

Open Source tools for Data Science

by IBM

About this Course

What are some of the most popular data science tools, how do you use them, and what are their features? In this course, you'll learn about Jupyter Notebooks, RStudio IDE, Apache Zeppelin and Data Science Experience. You will learn about what each tool is used for, what programming languages they can execute, their features and limitations. With the tools hosted in the cloud on Cognitive Class Labs, you will be able to test each tool and follow instructions to run simple code in Python, R or Scala. To end the course, you will create a final project with a Jupyter Notebook on IBM Data Science Experience and demonstrate your proficiency preparing a notebook, writing Markdown, and sharing your work with your peers. LIMITED TIME OFFER: Subscription is only \$39 USD per month for access to graded materials and a certificate.

Show less



Taught by: [Polong Lin](#), Data Scientist

IBM Developer Skills Network

Basic Info	Course 2 of 9 in the IBM Data Science Specialization
Level	Beginner
Commitment	3 weeks of study, 2-3 hours/week
Language	English, Subtitles: Russian Volunteer to translate subtitles for this course

How To Pass	Pass all graded assignments to complete the course.
User Ratings	Average User Rating 4.6

Syllabus

WEEK 1

Introducing Skills Network Labs

This week, you will get an overview of the various data science tools available to you, hosted on the IBM Developer Skills Network Labs (SN Labs). You will create an account and start exploring some of the features.

3 readings, 1 practice quiz

expandweek 1 material

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.

Graded: Quiz 1 - Skills Network Labs

Jupyter Notebooks

This week, you will learn about a popular data science tool, Jupyter Notebooks, its features, and why they are so popular among data scientists today.

2 videos

expandweek 1 material

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.

Graded: Quiz 2 - Jupyter Notebooks

WEEK 2

Apache Zeppelin Notebooks

This week, you will learn about Apache Zeppelin Notebooks, its feature, and how they are different from Jupyter Notebooks.

3 videos

expandweek 2 material

- 1.
- 2.
- 3.
- 4.
- 5.

Graded: Quiz 3 - Zeppelin Notebooks

RStudio IDE

This week, you will learn about a popular data science tool used by R programmers. You'll learn about the user interface and how to use its various features.

4 videos

expandweek 2 material

- 1.
- 2.
- 3.
- 4.
- 5.
- 6.

Graded: Quiz 4 - RStudio IDE

WEEK 3

IBM Watson Studio

This week, you will learn about an enterprise-ready data science platform by IBM, called Watson Studio (formerly known as Data Science Experience). You'll learn about some of the features and capabilities of what data scientists use in the industry.

4 videos, 2 readings

expandweek 3 material

- 1.

- 2.
- 3.
- 4.
- 5.
- 6.

Graded: Quiz 5 - IBM Watson Studio

Project: Create and share a Jupyter Notebook

1 reading

expandweek 3 material

- 1.

Graded: Create and share your Jupyter Notebook

[View Less](#)

How It Works

GENERAL

How do I pass the course?

To earn your Course Certificate, you'll need to earn a passing grade on each of the required assignments—these can be quizzes, peer-graded assignments, or programming assignments. Videos, readings, and practice exercises are there to help you prepare for the graded assignments.

What do start dates and end dates mean?

Once you enroll, you'll have access to all videos, readings, quizzes, and programming assignments (if applicable). If you choose to explore the course without purchasing, you may not be able to access certain assignments. If you don't finish all graded assignments before the end of the course, you can reset your deadlines. Your progress will be saved and you'll be able to pick up where you left off.

What are due dates? Is there a penalty for submitting my work after a due date?

Within a course, there are suggested due dates to help you manage your schedule and keep coursework from piling up. Quizzes and programming assignments can be submitted late without consequence. However, it is possible that you won't receive a grade if you submit your peer-graded assignment too late because classmates usually review assignment within three days of the assignment deadline.

Can I re-attempt an assignment?

Yes. If you want to improve your grade, you can always try again. If you're re-attempting a peer-graded assignment, re-submit your work as soon as you can to make sure there's enough time for your

classmates to review your work. In some cases you may need to wait before re-submitting a programming assignment or quiz. We encourage you to review course material during this delay.

Show less

PEER-GRADED ASSIGNMENTS

Peer-graded assignments require you and your classmates to grade each other's work.

How do peer graded assignments work?

After you submit your work, you'll be asked to review your classmates' assignments. To pass, you'll need to earn a passing grade on your submission and complete the required number of reviews.

How are grades calculated?

You and your classmates will be asked to provide a score for each part of the assignment. Final grades are calculated by combining the median scores you received for each section.

What kind of feedback should I give?

Be respectful, encouraging, and honest. Acknowledge what your classmate did well and offer specific suggestions on how they can improve. Scores should reflect the learner's understanding of the assignment prompt and points should not be deducted for difficulties with language or differences in opinion.

Is there a penalty for submitting my work late?

No, but it's important to submit your work as close to the due date as you can. Classmates grade most of the assignments within three days of the due date. If you submit yours too late, there may not be anyone to review your work.

If I fail an assignment, can I try again?

Yes! You can always try again, but you'll need to resubmit your work as soon as possible to make sure your classmates have enough time to grade your work.

Can I edit my assignment?

Yes, but you'll need to re-submit your work and any grade you've already received will be deleted.

Show less

Introducing Skills Network Lab Week-1

What is Skills Network Lab?

Welcome to Skills Network Labs, previously known as Data Scientist Workbench or BDU Labs. It's a free virtual lab environment that lets you practice your skills in Python, R, and more, while you learn data science.

Skills Network Labs is an environment that contains tools such as RStudio, Jupyter Notebook, and Zeppelin Notebook. These tools provide an interactive environment for you to perform data analysis, data visualization, machine learning and image recognition. For example, on Jupyter Notebook you can train a model that can recognize the denomination of an Euro banknote by using just the image of the banknote.

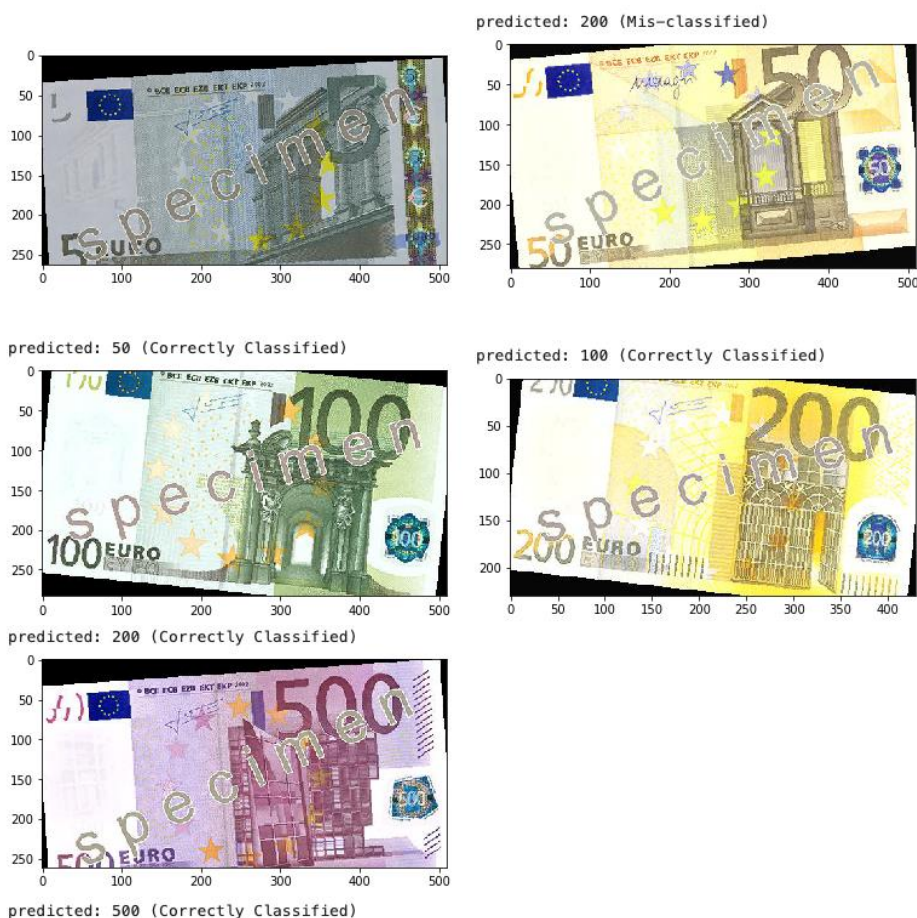
```
[16]: # Generate test dataset and generate the prediction results

test_valid_generator = ImageDataGenerator().flow_from_directory(validation_data_dir
                                                                , target_size=(224, 224)
                                                                , batch_size=5
                                                                , classes=classes
                                                                , seed=0
                                                                , shuffle=False)
```

Found 70 images belonging to 7 classes.

Create a test dataset using validation data. Plot 5 random images which index is in the `numbers` list. Also print the predicted label

```
[17]: # Answer for Question 3.3: Plot five random images and their predictions
```



It's ready to use in your web browser. So without having to install anything on your computer, you can immediately start using popular data science tools like RStudio IDE, Jupyter Notebooks, Apache Zeppelin and more. In these videos, you'll be learning about these popular tools and how to get familiar working in the environments. Skills Network Labs is also the virtual lab environment for cognitiveclass.ai, a popular website with lots of free online courses in data science, artificial intelligence, and data engineering. But throughout these videos, you'll also hear us referring to Data Scientist Workbench, which is simply the former name of Skills Network Labs. So in actuality, Skills Network Labs and Data Scientist Workbench are one and the same. Let's first go through some of the user interface on Skills Network Labs shown in the image below to get you familiarized with the environment before jumping into the data science tools.

Skills Network Labs Account ~ My Data

Manage Data
My Data

Build Analytics
JupyterLab
Zeppelin Notebook
RStudio IDE

Build Analytics on GPU
JupyterLab for IBM PowerAI
IBM PowerAI Vision

Deprecated
Jupyter Notebook
Jupyter for IBM PowerAI

Resources
Submit an idea
Blog
Knowledge Base
Feedback Forum
Support
Online Learning
What's new

What do you want to do today?

Manage Data	Prepare Data
My Data	Coming soon
Coming soon	

Build Analytics	Build Analytics on GPU
JupyterLab	JupyterLab for IBM PowerAI
Zeppelin Notebook	IBM PowerAI Vision
RStudio IDE	
Coming soon	

Deprecated
Jupyter Notebook
Jupyter for IBM PowerAI

Support

Learn more about what's in your Workbench

Zeppelin Notebook

- Tutorials
- How to import data for use by Apache Zeppelin? ↗

RStudio IDE

- Introduction to R on Cognitive Class ↗

Thank you for reading this article.

Account Features

Note: Skills Network Labs is a very dynamic environment and tools can be added and may also be removed at any time. In addition, as with any web application, the layout and placement of items can change at any time. As a result, you should pay attention to the intent of the lessons rather than strictly following instructions or details in the screenshots.

Welcome to Skills Network Labs which you can access with labs.cognitiveclass.ai. On the Skills Network Labs homepage, you have all the data science tools at your fingertips. Once you're on the main page of Skills Network Labs, you'll find several buttons. Let's take a closer look at each of these building analytics tools.

The screenshot shows the Skills Network Labs homepage. On the left is a dark sidebar with a menu. The main content area has a header 'What do you want to do today?' followed by four columns of buttons: 'Manage Data' (My Data, Coming soon), 'Prepare Data' (Coming soon), 'Build Analytics' (JupyterLab, Zeppelin Notebook, RStudio IDE, Coming soon), and 'Build Analytics on GPU' (JupyterLab for IBM PowerAI, IBM PowerAI Vision). Below these is a 'Deprecated' section with Jupyter Notebook and Jupyter for IBM PowerAI. At the bottom, there are two boxes for 'Zeppelin Notebook' and 'RStudio IDE' with links to tutorials. A 'Support' button is on the right.

Skills Network Labs Account ~ My Data

Manage Data
My Data

Build Analytics
JupyterLab
Zeppelin Notebook
RStudio IDE

Build Analytics on GPU
JupyterLab for IBM PowerAI
IBM PowerAI Vision

Deprecated
Jupyter Notebook
Jupyter for IBM PowerAI

Resources
Submit an idea
Blog
Knowledge Base
Feedback Forum
Support
Online Learning
What's new

What do you want to do today?

Manage Data
My Data
Coming soon

Prepare Data
Coming soon

Build Analytics
JupyterLab
Zeppelin Notebook
RStudio IDE
Coming soon

Build Analytics on GPU
JupyterLab for IBM PowerAI
IBM PowerAI Vision

Deprecated
Jupyter Notebook
Jupyter for IBM PowerAI

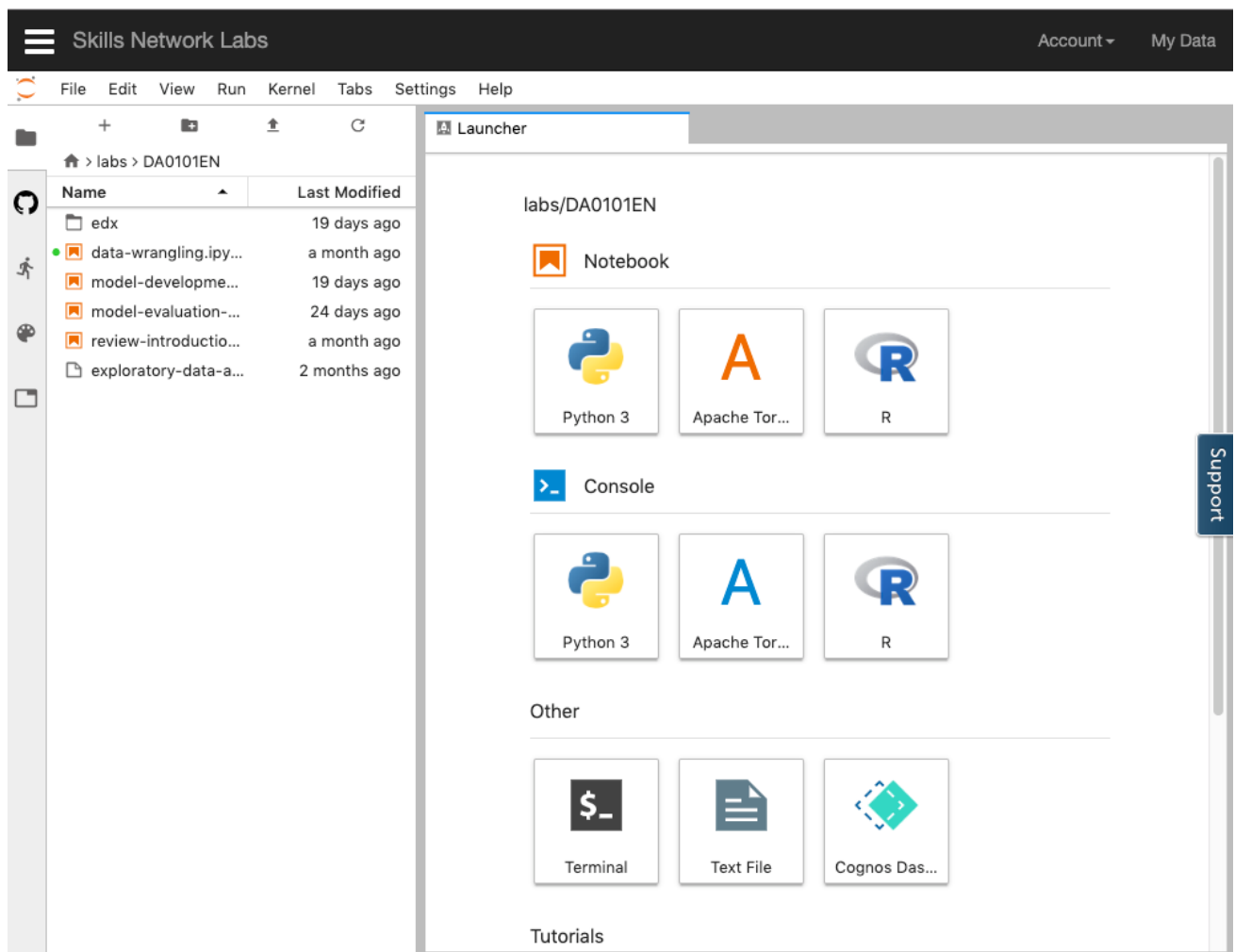
Support

Learn more about what's in your Workbench

Zeppelin Notebook
Tutorials
How to import data for use by Apache Zeppelin? ↗

RStudio IDE
Introduction to R on Cognitive Class ↗

JupyterLab button takes you to JupyterLab which is an open source tool that provides an interactive environment to run or create notebooks that run codes in Python with Jupyter Notebooks, Scala on Apache Toree, R and other programming languages. In addition to notebooks, JupyterLab provides, file browser to help you organize your files, console, terminal, GitHub integration and editors - functionality typically associated with Interactive Development Environments (IDE). JupyterLab is an extensible environment and Skills Network Labs is adding new functionality all the time.



Notebooks are a favorite tool of many data scientists. Let's take a look at a Python 3 notebook. Jupyter notebook is an interactive document that allows you to execute code in smaller chunks called cells. When you execute a cell, the notebook prints any output immediately into the output cell. Doing so, allows you to do a number of things. For instance, you can write your code to import data, print the data, clean the data, print the cleaned data, create a model and print the model output and so on. And you can change the code in an input cell and rerun the cell as often as you'd like, but that's not all, the notebook also supports rendering markup cells in line, so that you can embed text, markdown HTML images, videos and even interactive widgets, all within a notebook.

Skills Network Labs

Account My Data

File Edit View Run Kernel Tabs Settings Help

+

+

+

labs > DA0101EN

Name	Last Modified
edx	19 days ago
data-wrangling.ipynb	a month ago
model-developme...	19 days ago
model-evaluation-...	24 days ago
review-introductio...	a month ago
Untitled.ipynb	3 minutes ago
exploratory-data-a...	2 months ago

Untitled.ipynb

data-wrangling.ipynb

Python list "headers".

Python 3

[4]:

df = pd.read_csv(filename, names = headers)

Use the method `head()` to display the first five rows of the dataframe.

[5]:

To see what the data set looks like, we'll use the `head()` method.
df.head()

[5]:

	symboling	normalized-losses	make	fuel-type	aspiration	num-of-doors	body-style	drive-wheels	engine-location	wheel-base	...	engine-size	fuel-system
0	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi
1	3	?	alfa-romero	gas	std	two	convertible	rwd	front	88.6	...	130	mpfi
2	1	?	alfa-romero	gas	std	two	hatchback	rwd	front	94.5	...	152	mpfi
3	2	164	audi	gas	std	four	sedan	fwd	front	99.8	...	109	mpfi
4	2	164	audi	gas	std	four	sedan	4wd	front	99.4	...	136	mpfi

5 rows x 26 columns

As we can see, several question marks appeared in the dataframe; those are missing values which may hinder our further analysis.

So, how do we identify all those missing values and deal with them?

How to work with missing data?

Steps for working with missing data:

1. identify missing data
2. deal with missing data
3. correct data format


Identify and handle missing values

Let's look at another building analytics tool called Zeppelin Notebooks, Zeppelin notebooks allow Interactive Data Analytics. Like Jupyter, you use notebooks to ingest, discover analyze, visualize, and collaborate with your data.

Currently, Apache's Zeppelin supports many interpreters such as Apache Spark, Python, JDBC, Markdown and Shell.

Skills Network Labs

Account ▾My Data ▾

 **Zeppelin**


Notebook ▾Job


anonymous ▾


Welcome to Zeppelin!


Zeppelin is web-based notebook that enables interactive data analytics.
You can make beautiful data-driven, interactive, collaborative document with SQL, code and even more!


Notebook ↻


 Import note


 Create new note

 Zeppelin Tutorial

 My_Zeppelin_notebook

 Tutorial for Python

 Tutorial for Scala


 Trash


Help


Get started with [Zeppelin documentation](#)

Community

Please feel free to help us to improve Zeppelin,
Any contribution are welcome!

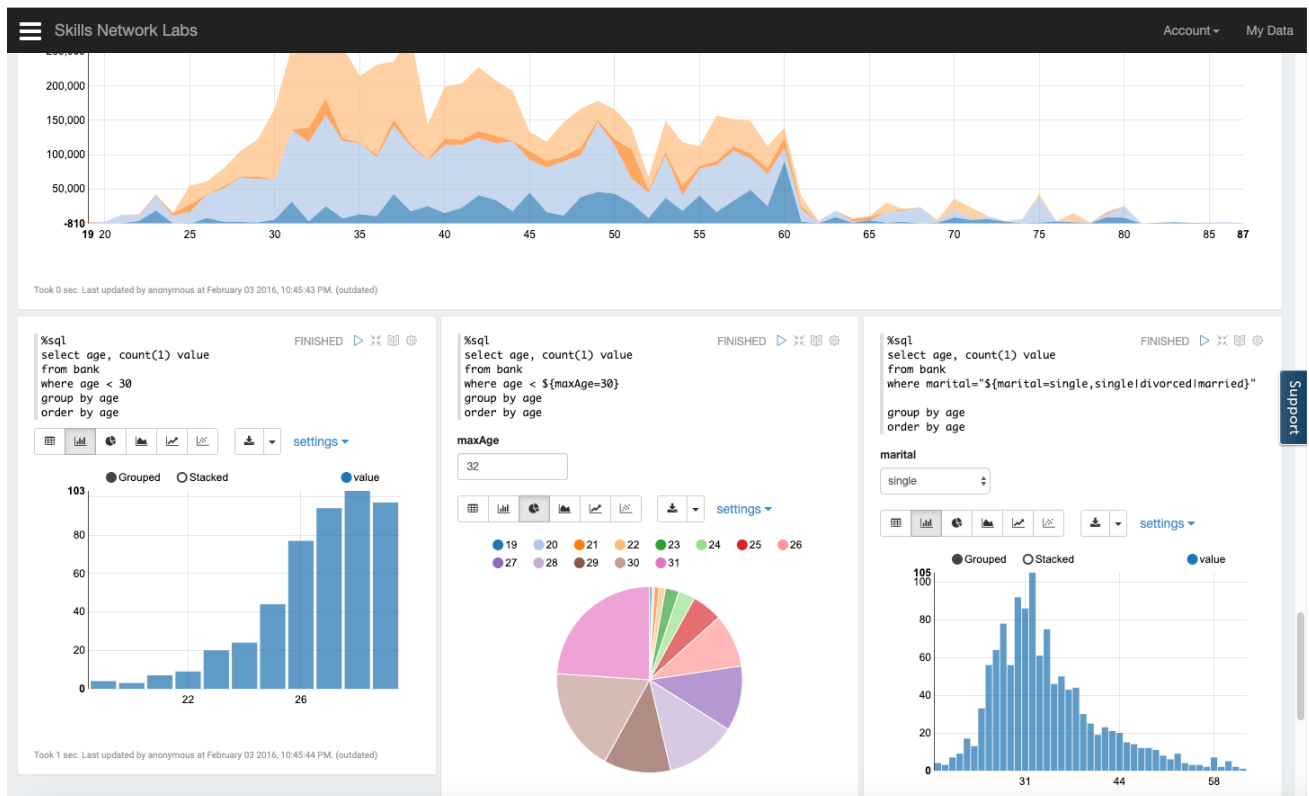
 Mailing list

 Issues tracking

 Github

Support

Let's look at Tutorial for Scala. Apache Zeppelin provides built-in Apache Spark integration so you don't need to build a separate module plugin or library for it. For data visualization, some basic charts are already included in Zeppelin allowing you to convert from data tables directly into visualizations without any code. Zeppelin also aggregates values and displays them in pivot charts, with simple drag and drop. You can easily create charts with multiple aggregated values including sum, count, average, minimum and maximum.



Finally let's look at RStudio in Building Analytics tools, RStudio IDE allows you to analyze data, take advantage of many statistical packages, create beautiful visualizations and Web applications. Like other IDEs, RStudio allows you to code in a console or a script editor as well as keep track of your variables and history. You can display your plots, manage your packages, and see help documentation for R. Taking it a step further, with R Shiny library, you can make your visualizations interactive. Using Shiny, you can create all sorts of Web based interactive apps just using R code.

Skills Network Labs

File
Edit
Code
View
Plots
Session
Build
Debug
Profile
Tools
Help

rstudio

Go to file/function

Addins

Console

Terminal

/resources/rstudio/

Environment

History

Connections

Import Dataset

Global Environment

Data

sc

Environment

Files

Plots

Packages

Help

Viewer

New Folder

Upload

Delete

Rename

More

Home

Name

Size

Modified

.Renvirom

52 B

Apr 6, 2019, 12:38 PM

.Rprofile

1.1 KB

Apr 6, 2019, 12:38 PM

addon-packages.lst

730 B

Aug 2, 2018, 2:56 PM

resources

Support

```

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

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.

#####

Welcome to RStudio IDE!

1. Your working directory is set to '/resources/rstudio'. We recommend you to use
'resources/data' to store your data that will be available to other Data Scientist Workbench
components, such as Jupyter Notebook.

2. SparkR run-time environment is initialized. The 'sc' SparkContext is already defined
for you to run Apache Spark from R.
#####

Attaching package: 'SparkR'

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

    as.data.frame, colnames, colnames<-, drop, endsWith, intersect,
    rank, rbind, sample, startsWith, subset, summary, transform, union

Spark package found in SPARK_HOME: /home/notebook/spark-2.3.1-bin-hadoop2.7
Launching java with spark-submit command /home/notebook/spark-2.3.1-bin-hadoop2.7/bin/spark-s
ubmit "--packages" "com.databricks:spark-csv_2.11:1.4.0" "sparkr-shell" /tmp/RtmpZyIUJg/bac
kend_port3a71c753390
> |

```

Thank you for reading this article.

Creating an Account

Welcome to Skills Network Labs. To get started with Skills Network Labs, you'll first need to create an account. Just go to labs.cognitiveclass.ai/register and sign up with your social media account.



Log in with social

or Log in with email

Click button to log in.



By logging in, I acknowledge that I understand how IBM Developer Skills Network is using my basic personal data, and that I am at least sixteen years of age. Our [Privacy Notice](#) provides more details.

Don't have an account? [Sign up!](#)

Lab - Getting Started with Skills Network Labs

This course uses Skills Network Labs, an online virtual lab environment to help you get hands-on experience with various data science tools without the hassle of installing and configuring the tools. You will get access to popular open-source data science tools right inside your browser, like Jupyter Notebooks and RStudio IDE.

Exercise 1 - Launching Skills Network Labs

1. To launch Skills Network Labs, click on the "**Open Tool**" button below.
2. Once Skills Network Labs loads, you should see the following screen:

The screenshot displays the Skills Network Labs web interface. On the left is a dark sidebar with navigation links: Manage Data (My Data), Build Analytics (JupyterLab, Zeppelin Notebook, RStudio IDE), Build Analytics on GPU (JupyterLab for IBM PowerAI, IBM PowerAI Vision), Deprecated (Jupyter Notebook, Jupyter for IBM PowerAI), and Resources (Submit an idea, Blog, Knowledge Base, Feedback Forum, Support, Online Learning, What's new). The main content area is titled "What do you want to do today?" and features four columns of tool buttons: Manage Data (My Data, Coming soon), Prepare Data (Coming soon), Build Analytics (JupyterLab, Zeppelin Notebook, RStudio IDE, Coming soon), and Build Analytics on GPU (JupyterLab for IBM PowerAI, IBM PowerAI Vision). A "Deprecated" section below shows Jupyter Notebook and Jupyter for IBM PowerAI. At the bottom, a "Learn more about what's in your Workbench" section highlights Zeppelin Notebook (with a link to tutorials and a guide on importing data for use by Apache Zeppelin) and RStudio IDE (with a link to an introduction to R on Cognitive Class). A "Support" button is visible on the right edge of the interface.

Fantastic! Now you're on your way to learning more about popular open data science tools.

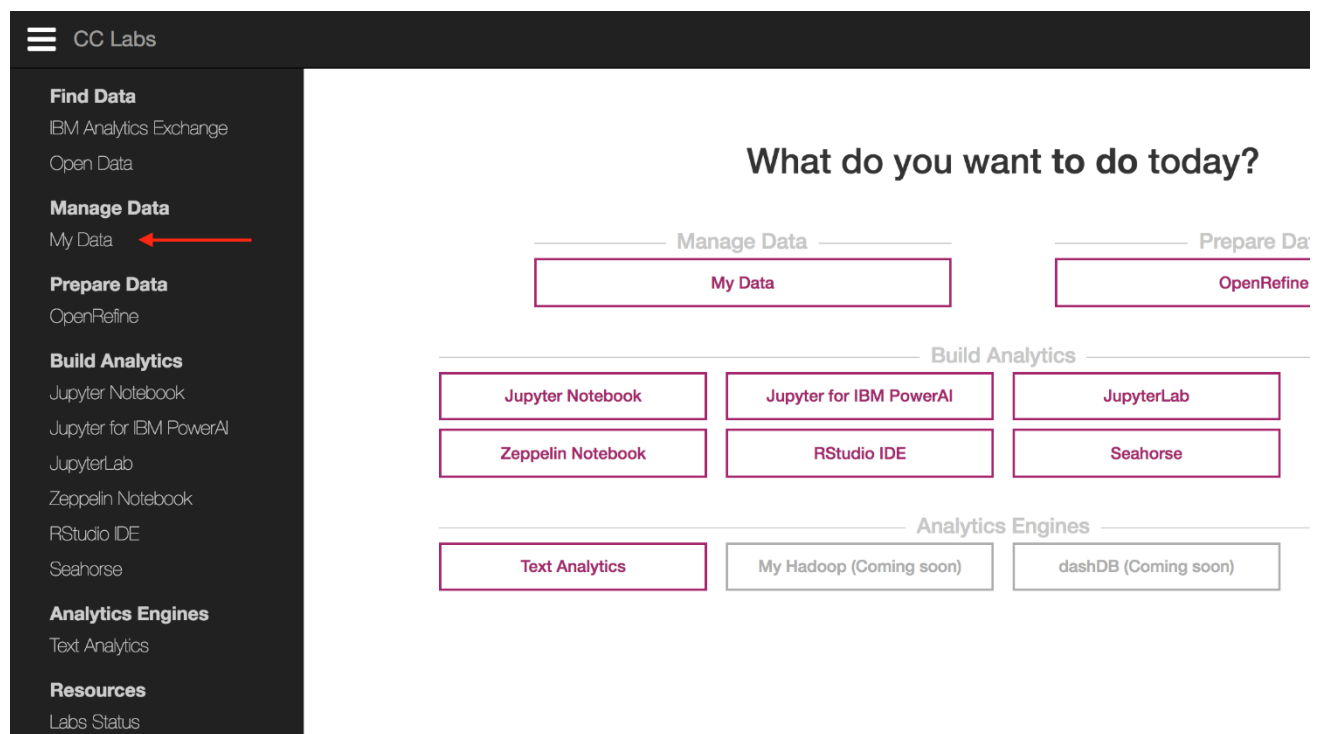
Please note that practice exercises are not graded, and you do not need submit anything.

This course uses a third-party tool, Lab - Getting Started with Skills Network Labs, to enhance your learning experience. The tool will reference basic information like your name, email, and Coursera ID.

Lab - Exploring My Data on Skills Network Labs

Exercise 1 - Using "My Data" on Skills Network Labs

1. If you already have Skills Network Labs open in a tab, you can skip to the next step. Otherwise, click on the "**Open Tool**" button below to launch Skills Network Labs. (Note: you may see errors if you have multiple tabs with Skills Network Labs open.)
2. Take a look at the **My Data** page. This is where you can upload and store your data for your account. It may take a few minutes to load.



2. Try **uploading a file**. If you don't know what to upload, you can use the following sample CSV file (student_grades.csv):

Sample CSV file:

[student_grades.csv](#)

3. Great! Now that you've uploaded the file, notice that you have a filepath at the top of the screen. You can use filepaths for any of your uploaded files to use with Jupyter Notebooks, RStudio IDE, and the other tools available on Skills Network Labs.

This course uses a third-party tool, Lab - Exploring My Data on Skills Network Labs, to enhance your learning experience. The tool will reference basic information like your name, email, and Coursera ID.

What are Jupyter Notebooks?

Welcome. In this video, we'll introduce you to Jupyter Notebooks.

Jupyter Notebooks are like documents, where you can execute chunks of programming code, one chunk at a time.

You can do everything from creating interactive maps, to creating interesting data visualizations, and even embedding videos.

Jupyter Notebooks are open-source and was designed for interactive data science in scientific computing.

Data scientists use Jupyter Notebooks, because in data science, you're often exploring your data or building models, and needing to see the outputs of parts of your code quite frequently, which Jupyter Notebooks enables.

Jupyter Notebooks were also designed to be shared with others.

You can tell stories in Jupyter Notebooks with your data by combining your code with explanatory text, output from your code, images, and videos.

Getting started with Jupyter Notebooks

Welcome, in this video we'll introduce you to JupyterLab which is an environment that allows you to easily edit and organize Jupyter Notebooks. Now you're going to hear the terms JupyterLab and Jupyter Notebook a lot.

So just to clarify here, when I'm talking about Jupyter Notebook, I'm referring to the actual file.

When I'm talking about JupyterLab, then that's the environment that organizes my various Jupyter Notebooks that allows me to run them.

You also learn how to upload data into JupyterLab, create your first Jupyter Notebook, run some code, and add some text and images.

When you open up JupyterLab for the first time, you'll see that on the left-hand side you have your files directory, where you can keep track of all of your files and a launcher screen on the right-hand side.

From this launcher, you can create new Jupyter notebooks currently available in Python 3, Scala, or R.

And there's some other options for more intermediate users to explore, by coding directly in the console using terminal or writing your code in a text editor.

On CC labs, there are other features like Cognose Dashboard embedded for data visualization and some sample Jupyter Notebook tutorials at the bottom.

For this video, we'll just focus on the Jupyter Notebooks.

Okay, let's first create a notebook in Python 3, and then we can dive into how to use Jupyter Notebooks.

Jupyter Notebooks are made up entirely of cells.

So, you can see in this notebook, there is currently one cell.

Let's type something inside the cell, like `1 + 1` and then on your keyboard press Shift+Enter, which executes the code using Python, and that returns two.

Now we can create even more cells by pressing the Plus button at the top, and write more code, like setting X to be equal to the integer one, and printing the output of X in the cell below.

So, all of these cells are actually what are called code cells, which allow me to run code using the interpreter which in this case is Python 3.

But how do we add titles or text to our notebook?

Well, you will first need to convert the cell type from code into a markdown cell by clicking on a cell choosing markdown from the drop-down menu.

And now if you type in something like one plus one into this markdown cell and try to run it using Shift + Enter, it gets converted directly into text. And to edit it again, simply double-click on a cell.

Now it's called markdown because it's actually a kind of syntax that allows you to stylize your text.

For example, you can create titles or headers by using the pound symbol and a space followed by some text like My Title.

There are other ways to stylize your text as well.

And here are some examples.

But I encourage you to look up a markdown formatting guide to find out all the different ways you can format your text.

In markdown cells, you can also use HTML as well.

For example, if you want to embed an image, you can use the image source tag to embed an image.

Now, I want to show you on JupyterLab, how to import data like a CSV file and use it in a Jupyter notebook.

To import data, you can simply drag and drop your data file directly into the files directory on the left hand side.

Once the upload is complete, it will show in the directory.

And it doesn't have to be CSV, it could be any file type.

You can also create different folders to organize all of your files, especially if you're going to be working with lots of notebooks or data files.

For CSV files, you can even double click on the file to open up a preview of its contents, which is really nice, but you can't really do much other than view the contents however.

So, if you want to analyze or manipulate the data, that's when you want to use Python, R, or Scala.

Let's create a new notebook running Python 3.

To import a CSV file in Python, we need to use a read CSV function from the pandas library.

So, the first step is to import pandas as PD.

Next, you can read in the file using the file path to your CSV file, and let's assign it to a variable called DF.

Now if you print the head of DF, it should show us the first five rows of the CSV file.

So now you've learned how to write and execute code in a Jupyter notebook and how to add text, format it, and embed images.

Learning how to analyze and manipulate the data is a bit out of the scope of this video, but hopefully you've enjoyed learning how to upload and import data in JupyterLab.

Lab - Jupyter Notebooks - The Basics

Special note about the lab environment:

These Jupyter notebooks labs now use the latest environment: **JupyterLab**, which uses the same .ipynb Jupyter notebook files.

Exercise 1 - Create a new notebook in Python

1. If you already have Skills Network Labs open in a tab, you can click on **JupyterLab on the main page**.
2. To create a new notebook, click on any of the languages under "Notebook".

Exercise 2 - Write and execute code

1. In your new empty notebook (from Exercise 1), **click within the gray code cell and write some code**, like "1 + 1" (without quotation marks).
2. Execute the code, by either clicking the Play button in the menu above the notebook, or by pressing Shift+Enter on your notebook.
3. You should see in the output, "2".
4. Try executing other code (try simple math operations).
5. Great! Now you know how to write and run code in Jupyter Notebooks

Exercise 3 - Create new cells

1. In your Jupyter notebook, first **click on any of the existing cells** to select the cell.
2. From the menu, click on **"Insert"**, then **"Insert Cell Above"** or **"Insert Cell Below"**.

3. Great! Now you know how to insert new cells in Jupyter Notebooks. Note you can use the keyboard shortcuts: **[a]** - Insert a Cell **A**bove; **[b]** - Insert a Cell **B**elow.

Exercise 4 - Create Markdown cells and add text

1. In your notebook, click on any code cell, and in the drop-down menu in the menu above, change the cell type from "Code" to "Markdown". As you'll notice, you cannot create Markdown cells without first creating cells and converting them from "Code" to "Markdown".
2. In the Markdown cell, write some text like "My Title".
3. To render the Markdown text, make sure the cell is selected (by clicking within it), and press Play in the menu, or Shift+Enter.
4. Your Markdown cell should now be rendered!
5. To edit your Markdown cell, double-click anywhere within the cell. Note you can use the keyboard shortcut: **[m]** - Convert Cell to **M**arkdown

Please note that practice exercises are not graded, and you do not need submit anything.

This course uses a third-party tool, Lab - Jupyter Notebooks - The Basics, to enhance your learning experience. The tool will reference basic information like your name, email, and Coursera ID.

Markdown Cheatsheet

Adam Pritchard edited this page on May 29, 2017 · [96 revisions](#)

This is intended as a quick reference and showcase. For more complete info, see [John Gruber's original spec](#) and the [Github-flavored Markdown info page](#).

Note that there is also a [Cheatsheet specific to Markdown Here](#) if that's what you're looking for. You can also check out [more Markdown tools](#).

Table of Contents

[Headers](#)

[Emphasis](#)

[Lists](#)

[Links](#)

[Images](#)

[Code and Syntax Highlighting](#)

[Tables](#)

[Blockquotes](#)

[Inline HTML](#)

[Horizontal Rule](#)

[Line Breaks](#)

[YouTube Videos](#)

Headers

```
# H1
## H2
### H3
#### H4
##### H5
##### H6
```

Alternatively, for H1 and H2, an underline-ish style:

```
Alt-H1
=====
```

```
Alt-H2
-----
```

H1

H2

H3

H4

H5

H6

Alternatively, for H1 and H2, an underline-ish style:

Alt-H1

Alt-H2

Emphasis

Emphasis, aka italics, with *asterisks* or *underscores*.

Strong emphasis, aka bold, with **asterisks** or **underscores**.

Combined emphasis with **asterisks and underscores**.

Strikethrough uses two tildes. ~~Scratch this.~~

Emphasis, aka italics, with *asterisks* or *underscores*.

Strong emphasis, aka bold, with **asterisks** or **underscores**.

Combined emphasis with **asterisks and underscores**.

Strikethrough uses two tildes. ~~Scratch this.~~

Lists

(In this example, leading and trailing spaces are shown with with dots: ·)

1. First ordered list item

2. Another item

··* Unordered sub-list.

1. Actual numbers don't matter, just that it's a number

··1. Ordered sub-list

4. And another item.

··You can have properly indented paragraphs within list items. Notice the blank line above, and the leading spaces (at least one, but we'll use three here to also align the raw Markdown).

··To have a line break without a paragraph, you will need to use two trailing spaces.··

··Note that this line is separate, but within the same paragraph.··

··(This is contrary to the typical GFM line break behaviour, where trailing spaces are not required.)

* Unordered list can use asterisks

- Or minuses

+ Or pluses

1. First ordered list item
 2. Another item
- Unordered sub-list.
1. Actual numbers don't matter, just that it's a number
 2. Ordered sub-list
 3. And another item.

You can have properly indented paragraphs within list items. Notice the blank line above, and the leading spaces (at least one, but we'll use three here to also align the raw Markdown).

To have a line break without a paragraph, you will need to use two trailing spaces.

Note that this line is separate, but within the same paragraph.
(This is contrary to the typical GFM line break behaviour, where trailing spaces are not required.)

- Unordered list can use asterisks
- Or minuses
- Or pluses

Links

There are two ways to create links.

[I'm an inline-style link](https://www.google.com)

[I'm an inline-style link with title](https://www.google.com "Google's Homepage")

[I'm a reference-style link][Arbitrary case-insensitive reference text]

[I'm a relative reference to a repository file](../blob/master/LICENSE)

[You can use numbers for reference-style link definitions][1]

Or leave it empty and use the [link text itself].

URLs and URLs in angle brackets will automatically get turned into links.
http://www.example.com or <http://www.example.com> and sometimes example.com (but not on Github, for example).

Some text to show that the reference links can follow later.

[arbitrary case-insensitive reference text]: https://www.mozilla.org

[1]: http://slashdot.org

[link text itself]: http://www.reddit.com

[I'm an inline-style link](#)

[I'm an inline-style link with title](#)

[I'm a reference-style link](#)

[I'm a relative reference to a repository file](#)

[You can use numbers for reference-style link definitions](#)

Or leave it empty and use the [link text itself](#).

URLs and URLs in angle brackets will automatically get turned into links. <http://www.example.com> or <http://www.example.com> and sometimes [example.com](#) (but not on Github, for example).

Some text to show that the reference links can follow later.

Images

Here's our logo (hover to see the title text):

Inline-style:

```
![alt text](https://github.com/adam-p/markdown-  
here/raw/master/src/common/images/icon48.png "Logo Title Text 1")
```

Reference-style:

```
![alt text][logo]
```

```
[logo]: https://github.com/adam-p/markdown-  
here/raw/master/src/common/images/icon48.png "Logo Title Text 2"
```

Here's our logo (hover to see the title text):



Inline-style:



Reference-style:

Code and Syntax Highlighting

Code blocks are part of the Markdown spec, but syntax highlighting isn't. However, many renderers -- like Github's and *Markdown Here* -- support syntax highlighting. Which languages are supported and how those language names should be written will vary from renderer to renderer. *Markdown Here* supports highlighting for dozens of languages (and not-really-languages, like diffs and HTTP headers); to see the complete list, and how to write the language names, see the [highlight.js demo page](#).

Inline ``code`` has ``back-ticks around`` it.

Inline code has back-ticks around it.

Blocks of code are either fenced by lines with three back-ticks `````, or are indented with four spaces. I recommend only using the fenced code blocks -- they're easier and only they support syntax highlighting.

```
```javascript
var s = "JavaScript syntax highlighting";
alert(s);
```
```

```
```python
s = "Python syntax highlighting"
print s
```
```

...
No language indicated, so no syntax highlighting.
But let's throw in a `tag`.

```
var s = "JavaScript syntax highlighting";
alert(s);
s = "Python syntax highlighting"
print s
```

No language indicated, so no syntax highlighting in Markdown Here (varies on Github).
But let's throw in a `tag`.

Tables

Tables aren't part of the core Markdown spec, but they are part of GFM and *Markdown Here* supports them. They are an easy way of adding tables to your email -- a task that would otherwise require copy-pasting from another application.

Colons can be used to align columns.

```
Tables	Are	Cool
col 3 is	right-aligned	$1600
col 2 is	centered	$12
zebra stripes	are neat	$1
```

There must be at least 3 dashes separating each header cell.
The outer pipes (|) are optional, and you don't need to make the raw Markdown line up prettily. You can also use inline Markdown.

```
Markdown	Less	Pretty
*Still* | renders` | **nicely**
1 | 2 | 3
```

Colons can be used to align columns.

| Tables | Are | Cool |
|----------|---------------|--------|
| col 3 is | right-aligned | \$1600 |

| Tables | Are | Cool |
|---------------|------------|-------------|
| col 2 is | centered | \$12 |
| zebra stripes | are neat | \$1 |

There must be at least 3 dashes separating each header cell. The outer pipes (|) are optional, and you don't need to make the raw Markdown line up prettily. You can also use inline Markdown.

| Markdown | Less | Pretty |
|-----------------|-------------|---------------|
| <i>Still</i> | renders | nicely |
| 1 | 2 | 3 |

Blockquotes

> Blockquotes are very handy in email to emulate reply text.
> This line is part of the same quote.

Quote break.

> This is a very long line that will still be quoted properly when it wraps. Oh boy let's keep writing to make sure this is long enough to actually wrap for everyone. Oh, you can **put** **Markdown** into a blockquote.

Blockquotes are very handy in email to emulate reply text. This line is part of the same quote.

Quote break.

This is a very long line that will still be quoted properly when it wraps. Oh boy let's keep writing to make sure this is long enough to actually wrap for everyone. Oh, you can *put* **Markdown** into a blockquote.

Inline HTML

You can also use raw HTML in your Markdown, and it'll mostly work pretty well.

```
<dl>
<dt>Definition list</dt>
<dd>Is something people use sometimes.</dd>

<dt>Markdown in HTML</dt>
<dd>Does *not* work **very** well. Use HTML <em>tags</em>.</dd>
</dl>
```

Definition list

Is something people use sometimes.

Markdown in HTML

Does **not** work ***very*** well. Use HTML *tags*.

Horizontal Rule

Three or more...

Hyphens

Asterisks

—

Underscores

Three or more...

Hyphens

Asterisks

Underscores

Line Breaks

My basic recommendation for learning how line breaks work is to experiment and discover -- hit <Enter> once (i.e., insert one newline), then hit it twice (i.e., insert two newlines), see what happens. You'll soon learn to get what you want. "Markdown Toggle" is your friend.

Here are some things to try out:

Here's a line for us to start with.

This line is separated from the one above by two newlines, so it will be a **separate paragraph**.

This line is also a separate paragraph, but...

This line is only separated by a single newline, so it's a separate line in the **same paragraph**.

Here's a line for us to start with.

This line is separated from the one above by two newlines, so it will be a *separate paragraph*.

This line is also begins a separate paragraph, but...

This line is only separated by a single newline, so it's a separate line in the *same paragraph*.

(Technical note: *Markdown Here* uses GFM line breaks, so there's no need to use MD's two-space line breaks.)

YouTube Videos

They can't be added directly but you can add an image with a link to the video like this:

```
<a  
href="http://www.youtube.com/watch?feature=player_embedded&v=YOUTUBE_VIDEO_ID_HERE  
" target="_blank"></a>
```

Or, in pure Markdown, but losing the image sizing and border:

```
[![IMAGE ALT TEXT  
HERE](http://img.youtube.com/vi/YOUTUBE_VIDEO_ID_HERE/0.jpg)](http://www.youtube.com/  
watch?v=YOUTUBE_VIDEO_ID_HERE)
```

Referencing a bug by #bugID in your git commit links it to the slip. For example #1.

License: [CC-BY](#)

Lab - Jupyter Notebooks - More Features

This lab is a continuation of the previous lab.

Note: If you have JupyterLab on Skills Network Labs open in a tab already, continue using that tab. Otherwise, click on "Open Tool" button below to launch Jupyter Notebooks.

Exercise 5 - Rename your Notebook

1. In the list of notebooks in the right-hand panel, click on the arrow (>) to the left of the notebook name to expand the list of options.
2. Click on "Rename" to rename your notebook to something like "My_Notebook.ipynb".

Exercise 6 - Save and Download your Jupyter Notebook from Skills Network Labs to your computer

Although your notebooks and data is preserved in your account after signing out, sometimes you may want to download your notebook to your computer.

1. To save, click on the Save button in the menu, or go to "Save Notebook".
2. To download the notebook, click on the Files pane to open the list of files. Navigate to your notebook, and right-click on the file to download the notebook.

Exercise 7 - Upload a Jupyter Notebook to Skills Network Labs

1. To upload a Jupyter Notebook from your computer to Skills Network Labs, drag and drop a .ipynb file from your computer directly into the browser. You can use the notebook you downloaded from Exercise 6.
2. Your notebook should open up automatically once it has finished uploading.

Exercise 8 - Change your kernel (to Python 3, R) in JupyterLab

1. With a notebook open, click on the kernel name (e.g., Python 3) in the top-righthand corner of the notebook to open up a pop-up window to change it to a different language.

This course uses a third-party tool, Lab - Jupyter Notebooks - More Features, to enhance your learning experience. The tool will reference basic information like your name, email, and Coursera ID.

Lab - Jupyter Notebooks - Advanced Features

This lab is a continuation of the previous lab.

Note: If you have JupyterLab on Skills Network Labs open in a tab already, continue using that tab. Otherwise, click on "Open Tool" button below to launch JupyterLab.

Exercise 9 - Advanced Markdown styling

1. You can write HTML code in your Markdown cells. Try executing the following HTML code in a Markdown cell: `Cognitive Class`. You should now see a hyperlink to `https://www.cognitiveclass.ai` that appears as "Cognitive Class".
2. You can also use Markdown formatting. For example, to use an H1 Header, try running the following in a Markdown cell: `# Header 1`. You can find more rules for Markdown formatting in this Markdown Cheatsheet: <https://github.com/adam-p/markdown-here/wiki/Markdown-Cheatsheet>
3. Using only Markdown, try creating a **table**, a **list** of shopping items, and try embedding an **image**.

This course uses a third-party tool, Lab - Jupyter Notebooks - Advanced Features, to enhance your learning experience. The tool will reference basic information like your name, email, and Coursera ID.

What are Zeppelin Notebooks?

Welcome, so what are Zeppelin Notebooks?

In this video, we'll be looking at Zeppelin notebooks, one of the open-source, web-based tools in Data Scientist Workbench. Zeppelin notebooks are multipurpose notebooks that can handle all your analytics needs, from data ingestion, data discovery, data analytics, to data visualization and collaboration. The Zeppelin interpreter concept allows any language or data processing backend to be plugged into Zeppelin. Currently Apache Zeppelin supports many interpreters such as Apache Spark, Python, JDBC, Markdown, and Shell.

Apache Zeppelin, in particular, provides built-in Apache Spark integration. You don't need to build a separate module, plug-in, or library for it. Apache Zeppelin with Spark integration provides a number of great features including automatic Spark context and SQL context injection. Runtime JAR dependency loading from local file system or Maven repository, as well as cancelling job and progress display.

For further information about Apache Spark in Apache Zeppelin, take a look at the Apache Spark in Zeppelin Notebooks video. For data visualization, here are some basic charts already included in Apache Zeppelin. Visualizations are not limited to Spark SQL query. In fact, any output from any language backend can be recognized and visualized. For pivot charts, Apache Zeppelin aggregates values and displays them in a pivot chart with simple drag and drop. You can easily create charts with multiple aggregated values including sum, count, average, minimum, and maximum. For dynamic forms, Apache Zeppelin can dynamically create some input forms for your notebook.

Apache Zeppelin is Apache 2.0 licensed software. Zeppelin notebooks are 100% opensource, so please check out the source repository and how to contribute. In fact, Apache Zeppelin has a very active development community. Please feel free to join our mailing list, and report issues if you'd like on Jira issue tracker. This brings us to the end of this video.

Thanks for watching.

Zeppelin for Scala

Welcome to Zeppelin for Scala.

In this video, we'll run through the Zeppelin tutorial for Scala, which reads data from a comma separated, .csv file, and uses Spark to convert it into a Spark DataFrame. Then we query data using a SQL command and visualize it.

To get started, just click the Zeppelin Notebook button on the main page. The Zeppelin welcome page opens. Then, click the Tutorial for Scala link. This launches the Notebook that we'll run through.

The first part of the tutorial describes the Interpreter Binding settings, namely for Spark, %spark is the default, md, angular, and sh. These are the kernels or interpreters that you select to be available in your Notebook. As a user, you can simply click on Save.

The first step in the tutorial downloads the bank.csv file. You'll need to import the packages that are needed for this tutorial.

The second step converts the.csv file to RDD by running the script.

The third step cleans the data using map and filter, which basically does three things. First, it creates an RDD of tuples from the original bank text. Second, it creates the schema or class represented to define the name and type of column. And third, it applies the schema to RDD. The fourth step creates the DataFrame using the toDF function.

A DataFrame is a distributed collection of data organized into named columns. It is conceptually equivalent to a table in a relational database, or a DataFrame in RPython, but with richer optimizations under the hood.

The fifth step registers the DataFrame as a temporary table.

The sixth step retrieves the data.

Now, the bank table can be easily queried with SQL commands.

And finally, the seventh step visualizes the data.

Some basic charts are already included in Zeppelin.

Please, feel free to create your own Notebook in the Notebook menu.

Getting some practice is always helpful.

This brings us to the end of this video. Thanks for watching.

Getting started with Zeppelin

Welcome to getting started with Zeppelin Notebooks.

Zeppelin Notebooks help you with all your analytics needs, including data ingestion, data discovery, data analytics, and data visualization and collaboration.

To get started with Zeppelin Notebooks on Data Scientist Workbench, once you're on the main page, just click on the Zeppelin Notebook button.

When the Zeppelin Welcome page opens, you'll find a number of links on the left that work with the notebook.

The links on the right point to Zeppelin Documentation and the Community.

Let's begin by focusing on what can be done with notebooks.

First, to import a note, just click the Import Note link. By default, the name of the imported note is the same as the original note, but you can override it by providing a new name.

To create a new note, click the Create New Note link. Use the generated name or type a new name and click Create Note. When you do this, a new empty Zeppelin Notebook displays.

Let's move on to note layout. Each Zeppelin note is composed of one to end paragraphs. The note can be viewed as a paragraph container. Each paragraph consists of two

sections. Code Section is where you put your source code, and Result Section is where you can see the result of the code execution.

On the top right corner of each paragraph, there are some commands that will allow you to execute the paragraph code, Hide/Show Code Section, Hide/Show Result Section or configure the paragraph. To configure the paragraph, just click on the gear icon.

From this dialog, you can, in descending order, control paragraph with, and since Zeppelin is using the grid system of Twitter Bootstrap, each paragraph width can be changed from 1 to 12. You can move the paragraph one level up, move the paragraph one level down, create a new paragraph, change paragraph title, show/hide line number in the code section, disable the Run button for this paragraph, export the current paragraph as an eye-frame and open the eye-frame in a new window, clear the result section and delete the current paragraph.

At the top of the note, you can find a toolbar which exposes command buttons, as well as Configuration, Security, and Display Options.

They are, Execute all the paragraphs sequentially in their display order, Hide/Show code section of all paragraphs, Hide/Show result section of all paragraphs, Clear the results section of all paragraphs, Clone the current note, Export the current note to a JSON file.

Please note that the code section and the result section of all paragraphs will be exported.

If you have heavy data in the result section of some paragraphs, it's recommended to clean them before exporting.

Delete the note and schedule the execution of all paragraph using a Cron syntax.

On the right side of the note toolbar, you can find configuration icons, display all the keyboard shortcuts, configure the interpreters binding to the current note, and switch the No display mode between Default, Simple and Report.

This brings us to the end of this video.

We hope you'll take the time to explore Zeppelin Notebooks. Thanks for watching.

What is RStudio IDE?

Welcome to "What is RStudio IDE?"

RStudio IDE is one of the open source web-based tools in Data Scientist Workbench.

It allows you to analyze data, take advantage of many statistical packages and create beautiful visualizations and web applications.

As its name suggests, RStudio IDE is only for R. Like other IDEs, in RStudio IDE, you can code in a console or script editor, keep track of your variables and history, display your plots, manage your packages and see help documentation for R. If you're an R user, you'll already be familiar with this type of setup, where you can analyze data, see your console and visualize your plots and graphs all in the same environment.

Another great feature is that you'll also have access to Apache Spark. By simply typing SC from the editor or console and running it, you'll see that you can run Spark R code to analyze big data.

Taking it a step further, R also gives you access to a Shiny library, which you can use to make your visualizations interactive. Using Shiny, you can create all sorts of web-based interactive apps just using R code.

To open RStudio IDE, simply click the RStudio IDE button on the Data Scientist Workbench home page.

This brings us to the end of this video. Thanks for watching.

Uploading files, installing packages and loading libraries in RStudio IDE

Welcome to RStudio Installing and loading libraries.

In this module, we'll focus on the bottom right quadrant of RStudio IDE.

On the files tab, you can create a new folder or upload files from a selected directory. Just remember to click, "choose file" to specify the directory path of the file to be uploaded. Or, using the "More" drop-down, you can copy, move, export files, and set or navigate to a working directory.

Let's say that you wanted to upload an Excel file from your local computer into the work-space. Simply click the upload icon in the files tab and you'll get a message that the file is uploading. And once the upload finishes, you can see it in your work-space.

And let's upload an R-script from RStudio resources studio folder into the work-space.

Click on resources shown here, then click on the resources link, then on our studio folder and here are the R-script files we've created. Clicking the R-folder you want, displays the code in R-editor on the top left quadrant.

So now you know how to upload data in R-scripts from Rstudio IDE or other types of files from your local or external drive. To check your working directory, type, getwd. Getwd returns the current working directory.

It's important to note that the working directory is always displayed on the console header as well.

The packages tab shows you all the packages that are installed. You can install other R-packages using the install icon. Otherwise, you can run a specific command in R to install a package via code. This command is `install.packages` with the package name within double quotes.

This concludes our video in which we reviewed uploading files, installing packages, and loading libraries. We encourage you to practice what you've learned to capture the full potential of RStudio IDE. Thanks for watching.

Getting started with RStudio IDE

Getting started with RStudio IDE.

In this video, we'll be reviewing how to use RStudio IDE. You'll also see how you can use Spark.

From the Data Scientist Workbench home page click RStudio IDE to open the tool.

First, let us talk about the interface. On the left-hand side there is the console where you can run commands and see your output. For example, let's assign the value one to `x`, as you can see here on the bottom left.

Then, after pressing enter, we can see what `x` is, under the Environment tab on the right side. Or I can type one plus one and press enter to get the sum directly on the line below.

It's probably better to write code in a text or R Script editor, which recognizes and corrects our syntax. Then, just run the R code commands from the script editor instead.

So let us create the editor. Select file, new file, R Script. This shifts the console to the bottom left quadrant and opens the R editor in the top left quadrant, where you can enter your R code.

Let's type one plus two from the R editor. As you can see, both the commands and the corresponding output are displayed in the console when you press enter or run the code.

It's important to note that in the R editor toolbar you have icons that will allow you to work faster.

Of the icons shown here, the most important one you'll want to know is the save button, which allows you to save your R Script to your Workbench account.

Please note that the R editor toolbar is not available in the console.

On the right side of the R editor you'll see two important buttons, they are the run and rerun buttons. As the name suggests, the run button allows you to run the current line or selection. To do so, simply press control enter if you are using a PC, or command enter if you are using a Mac. To rerun the previous code region press control shift R on a PC, or command shift R on a Mac.

In the lab you'll get to familiarize yourself with these icons, including learning more about and practicing with the powerful code tools dropdown menu.

Coding in R has never been easier.

So, to run your code, selected lines or the current line, and have the results display in the console, you simply click the run icon on the toolbar or press control enter on a PC, or command enter on a Mac.

You can run the same R code as many times as you like.

So, if you add another line in the editor, for instance one plus 1000, you can highlight both lines and run them. By default, the editor runs only the current line. However, if you highlight multiple lines it will run all the code that is highlighted and the console displays this.

Note that the current directory path displays next to the console window title.

When you click the arrow beside the directory the files tab is refreshed on the bottom right. After saving the current code, the file name displays in that directory, in the files tab, located in the bottom right quadrant.

To save the current code, click file save, which opens up the save file window. Or you can click on the save button at the top of the script. Now, simply enter the file name and click save.

Click the different tabs in this quadrant to see plots of your data, to see your packages, to get help, and to view your data.

We encourage you to play with RStudio IDE to learn about and leverage its full potential.

This is the end of this getting started video. Thanks for watching.

RStudio Environment and History

Welcome to RStudio IDE, environment and history.

As you see here, the top right quadrant of Rstudio IDE contains the Environment and History tabs.

The Environment tab contains a list of all the variables you've used in your R code.

So when we said, x is equal to 1 in the console, the variable x and its current value displays in this tab. This is a good way to quickly see and keep track of all the variables you've defined.

Note that the SC environment variable is for Apache Spark, which is pre-installed for you. If you don't need it or want to remove it, you can do so from the environment using the command rm, as in rm with sc within parentheses.

Another interesting feature within RStudio IDE is that it includes built in datasets, which in R are called a data frame. An example of such a data set is MTCars, which contains data on the latest motor trends. So if you were to type in mtcars in the R editor and run it, the output would display in tabular form, as shown here in the console.

There are other built-in data sets, and you can run the `data` command with a set of `md` parenthesis to see the full list. For example, you can type in the first one, `air passengers`, and run it to see the contents.

Now let's look at what's available and what's displayed on these two tabs and how to navigate in them. If you want to save all the variables in your R environment, you can click on the save button to save your workspace to disk as an R data file. This way, if you load the R data file, it'll load all of your saved variables and data frames back into R.

You can also import your own data sets by clicking on Import Data Set. The Import Data Set allows you to load files from your local drive or from a web URL.

And the last icon allows you to delete all objects from the current environment. Doing this as useful if you want to clean up your environment and start from an empty workspace again.

However, please be careful and use discretion with this action because it can not be undone.

The Global Environment's drop-down allows you to look at all the packages that are currently loaded in your environment.

When you click on the History tab, you'll be able to see all the code you've executed, but only in this session either from the console or from the source location of the data.

You can double click on an entry to paste it into console.

This ends the video on environment and history.

We encourage you to take some time to practice what you've learned, as well as doing the lab.

Thanks for watching.

Lab - RStudio - The Basics

Exercise 1 - Running your first R code in RStudio IDE

1. To launch **RStudio IDE** on Skills Network Labs, click on the "**Open Tool**" button below.
2. With RStudio IDE open in your browser, click within the **Console** window in the lower lefthand corner.
3. Type in "**1 + 1**", without quotation marks and press **Enter** to run the R code.
4. You should see the result: "**[1] 2**". Note that the "[1]" simply refers to the [n]th item returned in the output.
5. Now try running "**1:1000**" in your console, which should display the first 1000 integers from 1 to 1000.

Exercise 2 - Create a new R script

1. Go to **"File"** then **"New File"** then **"R Script"**.
2. In your R script window, type **"x <- 1"** or **"x = 1"**. Both do the same thing, but in R, variables are traditionally assigned using the **"<-"** syntax, which resembles a leftward-pointing arrow.
3. **Execute the code from the script window**, either by pressing the **"Run"** button or by pressing **"Ctrl+Enter (Windows)"** or **"Cmd+Enter (Mac)"** on your keyboard. You should see that the code was properly run in the console.
4. Notice that in the upper-righthand window under **"Environment"**, you should now see a new variable for **"x"**, with a value of **"1"**.
5. Try adding **100 to x** from the script window and confirm the result in the Console.

Exercise 3 - Save your new R script

1. Go to **"File"**, then **"Save"** to save your R script. Alternatively, you can go to **"File"** then **"Save As..."** to choose a filename and where you want to save your file.
2. In the bottom right-hand window for **"Files"**, you should now be able to navigate the **folders** and find your current R Script.
3. If you wish, you can **download the R script to your computer**. To do so, check the checkbox to the left of the filename, and click on **"Export"** to download your file.

This course uses a third-party tool, Lab - RStudio - The Basics, to enhance your learning experience. The tool will reference basic information like your name, email, and Coursera ID.

Lab - RStudio - Creating an interactive map in R

Note: If you have Skills Network Labs open in a tab already, continue using that tab. You can navigate to RStudio IDE by click on **"RStudio IDE"** in the menu (click on the three lines in the top-left corner). Otherwise, click on **"Open Tool"** button below to **launch RStudio IDE**. If you have two tabs or windows open with RStudio IDE, only the most recent tab/window will be active.

Exercise 4 - Upload an R script, install an R library, and plot an interactive map in R of New York City

1. Download the following **R script**:

[map_new_york_city.R](#)

2. Upload the R script to RStudio IDE by navigating to the **"Upload"** button in the lower-righthand window under the **"Files"** tab.

3. Once upload is complete, **click on the map_new_york_city.R script in RStudio** to open it in the script editor window.

4. **Try running each line of code**, one by one from the top, using "Run" or by pressing "Ctrl+Enter (Windows)" or "Cmd+Enter (Mac)" on your keyboard. Don't worry too much about what the code means or how to write similar code.

5. Once the map is created, you should see it in the lower right-hand corner. You can try to zoom in and out of the map view.

6. (Optional) Try changing the co-ordinates of the map (its latitude and longitude values) in the script window, then re-run the code again to re-create the map. For example, you can do a Google search for a popular landmark in the world to find its latitude and longitude. You may need to figure out which lines of code you *need* to change, which lines of code you *need* to re-run, and which lines of code you *do not need* to re-run.

This course uses a third-party tool, Lab - RStudio - Creating an interactive map in R, to enhance your learning experience. The tool will reference basic information like your name, email, and Coursera ID.

IBM Watson Studio

Week-3

What is IBM Watson Studio?

IBM Watson Studio, formerly known as Data Science Experience or DSX, is an enterprise-ready environment for data scientists and developers, and includes some of the tools as you have learned so far on Cognitive Class Labs. You may find that many of the features are similar as what you have seen on Cognitive Class Labs. However, because IBM Watson Studio was designed for scalability and enterprise usage, you will find some extra features that include (1) collaboration with team members, (2) scalability with Spark clusters to analyze big data, and (3) connections to various data sources.

IBM Watson Studio also comes with a free trial, which includes Jupyter Notebooks, RStudio, space for object storage, 2 Spark executors, and a community that includes notebooks and tutorials that you can use.

Note that the videos in this course may still include references to "Data Science Experience" or "DSX", but in your mind, simply replace those terms with "Watson Studio". Thank you for your patience with us as we update the videos!

You can register for IBM Watson Studio here:
https://cocl.us/Watson_Studio_Coursera_DS0105

What is IBM Watson Studio (Data Science Experience)?

Data Science Experience or DSX is an environment that brings together everything that a data scientist needs to be more productive, including tools, data, and content.

DSX is built on a foundation of Jupyter Notebooks, which enables data scientists to combine code execution in Python, R, or Scala with rich text, mathematics, plots, and rich media.

Data scientists can also leverage different tools to perform their analyses such as R studio, a popular tool among R programmers.

Data Science is also about collaboration, working in teams with other data scientists, developers, or business analysts.

On data science experience, you can share your work with others on social media, collaborate with others on your Jupyter Notebooks, and even do a version control with GitHub.

To get started on data science experience, you can visit the link below this video to sign up for a free account.

Creating an account on IBM Watson Studio (Data Science Experience)

In this video, I'll walk you through the process of signing up for an account on IBM Data Science Experience.

Data Science Experience uses services provided by IBM Cloud.

So, to get started, click on the IBM Cloud link below this video to register for your free IBM Cloud account.

Feel free to pause the video now and come back when you see this screen.

Okay, now that you're on this page, in the bottom right hand corner, click on the blue button for a sign up to create.

Next, you'll need to fill out this registration form. Make sure to use a valid email address because you will need to click on a confirmation email to activate your account. You can pause the video now and finish up the registration.

Okay, I will assume you've successfully activated your account. You should now see a Create button which will add Data Science Experience to your account. If you don't see this screen, click on the link below the video again and make sure to log in.

And at this point, go ahead and click on Create. Excellent. Now, to start Data Science Experience, click on the Get Started button.

Next, it will ask you which organization in space to create your Data Science Experience in your IBM Cloud account.

Don't worry, just keep the default settings and hit Continue.

Great. Your account is now created. All right.

Go ahead and take 30 seconds to explore the interface before we move on. Thank you.

Jupyter notebooks on Watson Studio (Data Science Experience) - Part 1/2

In this video I'll show you how you can begin creating and using a Jupyter notebook on Data Science Experience.

To create notebooks, you must first create a project to place your notebooks. So, let's get started.

There are different buttons to create a new project on the main page. But I like to always click on the project's button in the menu bar at the top of the page. And this way I can take a look at all of my projects. So, I'll go ahead and click on All Projects.

If you just signed up for DSX then you won't have any projects, like you see here.

So, go ahead and click on New Project, and remember we're creating a project to place our notebooks in. So, you'll need to first give a name to your project. Like my data science project and write a project description if you like.

Your project which will include your notebooks and other assets also need to point to two resources: in object storage where your project will be stored in the cloud, and also Spark executors to run your notebooks.

By the way these Spark executors will by default allow you to run Python R or Scala.

For object storage you can choose from any of the options, but here we'll use IBM cloud object storage which provides five gigabytes of cloud object storage for free.

Click on Add which sends you to the cloud object storage page.

Scroll down to the lite plan, the free plan and click on Create, then Confirm.

Back to the project creation page refresh the page to see the cloud object storage instance you just created.

Next, we need to add the Spark service to add to the project so that our notebooks will be able to run Python R, Scala and of course Spark.

Click Add, then again choose the lite plan which includes two Spark executors for free, then click Create and then Confirm.

Click Refresh on your project creation page in order to see the Spark service and now you can click on Create to create your project.

Now you'll see your main project page, but we'll just jump right into the assets page where you'll create or place your notebooks.

Your assets consist of data, notebooks and other items you add to your project.

Let's create a new Jupyter notebook and add it to this project.

To do so, click on New notebook.

Now, you can create a blank notebook or you can upload a notebook which is a dot IPYNB file from your desktop or you can import a notebook from a URL as long as it ends with dot IPYNB.

Let's go back to the blank tab to create an empty notebook.

Let's give our notebook a name like my first notebook.

The description is optional and you can choose between Python two, R, Scala or Python three.

If you know which Spark version you prefer using you can select that.

Otherwise, leave it as the default value and go ahead and click on Create Notebook, great.

Jupyter Notebooks on IBM Watson Studio (Data Science Experience) ***- Part 2/2***

Great. So, this is an empty Jupiter notebook.

You'll notice in the top right-hand corner you'll see, "Kernel starting, please wait," in a black circle next to the interpreter you've chosen, which means that the interpreter is busy and you won't be able to run or execute any code just yet.

There, now, the circle is a white empty circle, which means your interpreter is ready.

Okay, let's try writing some code here.

So, in this gray code cell here, I can type in something like one plus one and I can click on the button here and that says Run to run the code.

And what this does is it runs against the interpreter here Python 2 with Spark 2.1.

I can also type in something like x is equal to Hello World and then print the contents of x.

Instead of clicking on the Run button, I can also press Shift + Enter to execute the code cell.

If you want to add a new cell, you can click on any of the cells from the left-hand side, click on Insert and you can choose to Insert Cell Above or Below.

Here I'll add in a Cell Above and what I'll do is add a little title for my notebook.

So, here I want to add in some text, not code.

So, I can write something like My Title and what I can do is change the format from Code to Markdown.

And then from here, you can simply click within the Markdown cell and press Shift + Enter to convert that into its proper text or Markdown format.

And if you'd like to edit this again you can just simply double click within the cell and change it however you like.

Press Shift Enter and it will render.

There are many different ways to style your text.

For example, well let me first convert this to Markdown so you can see.

You can add in any kind of HTML text including images here.

You can change the size of your text using the Markdown formatting.

You can also add in lists, numbers, and tables and here you can also of course make your text Bold or Italics.

So, go ahead and run this so you can see the effect.

And just know there's one thing that's a little bit unintuitive, and that's this part here you'll see that I've written out Bold and Italics.

And originally these were on two separate lines.

See Bold and Italics are there.

So, to make sure that Bold and Italics are actually appearing in separate different lines here this is on the same line to appear as separate lines simply add in two empty spaces after each of these lines.

And that way that will ensure that the next line, Italics will be on its own separate line.

So that's a little bit unintuitive, I thought it'd be good for you to know.

So, I think those are the key things that you need to know for notebooks on IBM Data Science Experience.

You can create other notebooks for your projects.

You can save your notebooks as well by going to File, Save and Save Version, but just note that these notebooks will save automatically every two minutes, so you don't have to worry too much about saving.

You can also download these notebooks if you like to your local desktop.

And if you ever get stuck with your Kernels, you can go ahead and click on Kernel here in this menu.

You can click on Interrupt or you can click on Restart, which will restart the Kernel.

And if you do want to switch your interpreters from Python to R for example, you can go ahead and click on Change Kernel and choose the language that you want to work with.

Note that you can only use one language at a time, so you can only use R or you can only use Python 2 or Python 3 in the same notebook.

And just one last step is that if you like a little bit more real state so you can see more of your notebook, you can click on this button in the right hand corner and that will free up a little bit more of the space here so you can see more of your notebook.

So that's all for this video.

Happy coding with Jupiter notebooks on Data Science Experience.