

# Predicting Urban Growth using Apache Spark

---

By Zachery Slocum

This presentation is a limited overview of my project.

Please consult the webpage for more detail:

<https://freezurbern.github.io/ITCS8190-CourseProject/>

# Overview

---

- Research question: Can we predict urban growth using population, road geometry, and land cover?
- Answer: Yes, but population and road geometry are much better than land cover.
- Using multiple linear regression
  - Independent variables: land cover, population density, road density from **2011**
  - Dependent variable: urban land cover from **2016**
- Running on Google Cloud Platform's Dataproc service

# Beyond Linear Regression

---

- Distributed Prediction
  - Apache Spark DataFrames using Python
  - Using columns for each step of the equation
  - Calculations for each census tract across cluster
- Data Pipeline using R and ArcGIS Pro
  - R:
    - Census tract population estimates and geometry
    - primary/secondary road geometry
  - ArcGIS Pro:
    - National Land Cover Database raster image → CSV

# Cool things I learned

---

- Plotting images in a notebook running in the cloud
- Keep data safe in a Google Cloud Storage 'Bucket'
  - Accessible by cluster but not stored in the cluster
- GitHub Pages for documentation

# Results

Dependent variable from 2016

GEOID	roaddens	popdens	barren	water	nature	agric	urbany	cy
1001020100	0.00678981352	0.00017956058	0.00200455581	0.00145785877	0.65676537585	0.10141230068	0.24036446469	0.24816192176
1001020200	0.00632959225	0.00064757175	0.00215749730	0.00080906149	0.34169363538	0.01672060410	0.64077669903	0.63979962938
1001020300	0.01908926044	0.00062276864	0.00903010033	0.00000000000	0.31739130435	0.09080267559	0.59180602007	0.60003224988
1001020400	0.01431175569	0.00068433141	0.00000000000	0.00014234875	0.30306049822	0.10804270463	0.58875444840	0.59407910920
1001020500	0.00594801379	0.00089086815	0.00015665387	0.00062661549	0.16417325918	0.15485235372	0.68034777160	0.67868136601
1001020600	0.00809658603	0.00042732944	0.00256267409	0.00055710306	0.32345403900	0.14896935933	0.52701949861	0.52894433391
1001020700	0.00484186922	0.00011919023	0.01676888440	0.03505162643	0.50512382579	0.26721527832	0.19260926947	0.19998295296
1001020801	0.00107152947	0.00002342970	0.00222291712	0.04794567109	0.61262100283	0.28207934523	0.05735398086	0.06520516270
1001020802	0.00115896453	0.00005584656	0.00054532804	0.00434852103	0.73789465767	0.18120874781	0.07654807349	0.08438610733
1001020900	0.00070723386	0.00001892776	0.00160465873	0.00395488546	0.73950835466	0.20842154354	0.04811521634	0.05622133741

Independent variables from 2011

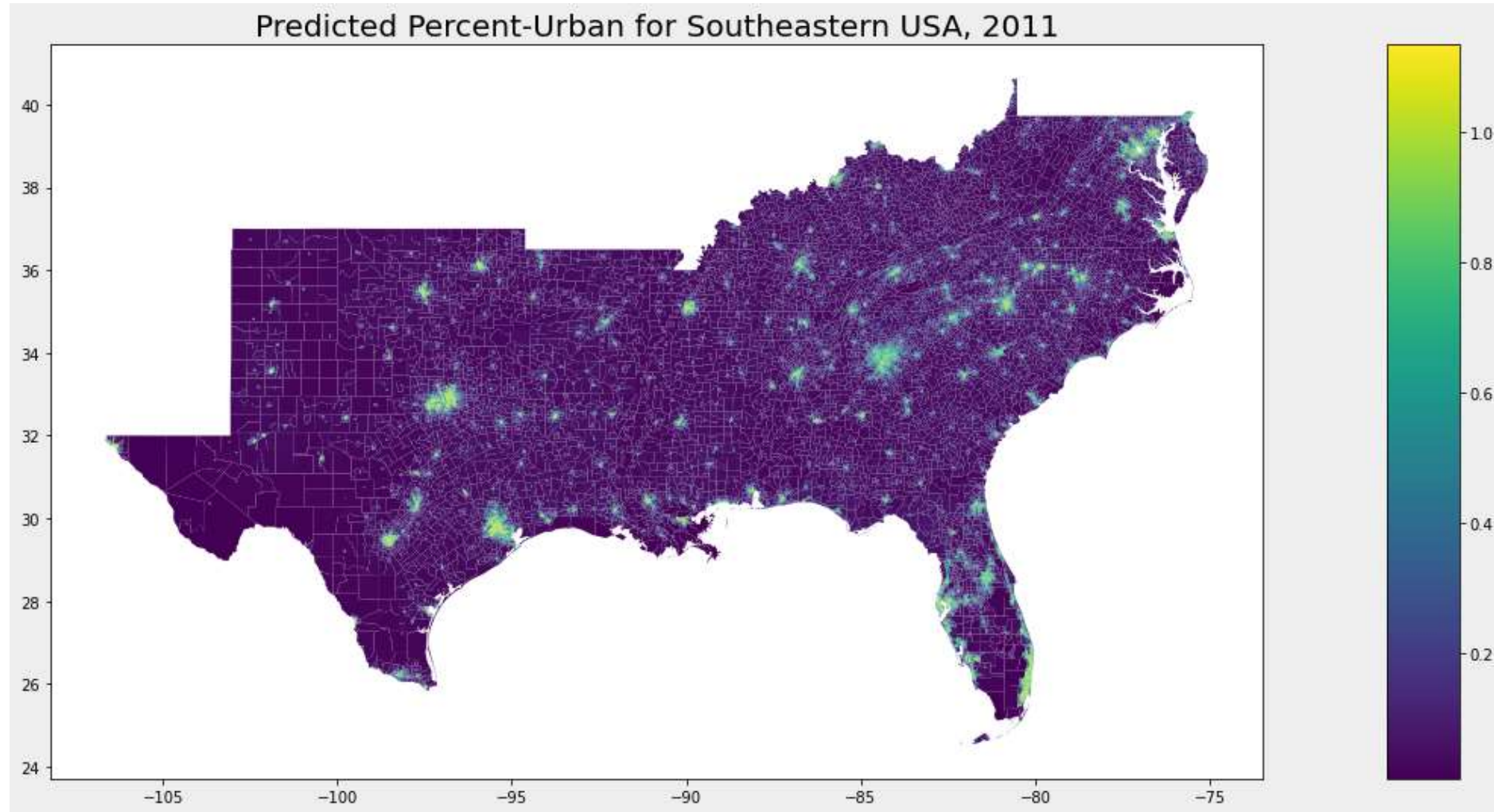
Note:

- Each row is a census tract (n=26,129)
- GEOID is the primary key (text)
- Yellow are independent variables (0..1)
- Purple is the actual value of the dependent variable (0..1)
- Green is the predicted value of the dependent variable (0..1)

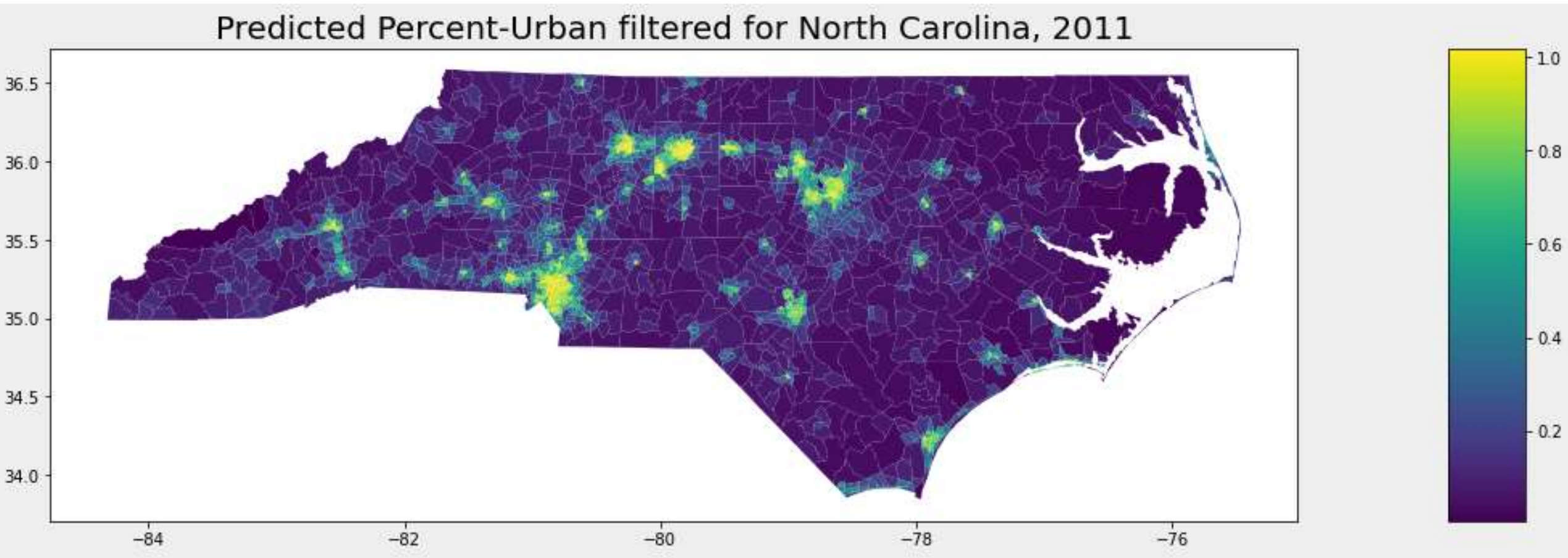
Beta Values	
intercept	0.9828573
roaddens	0.6322004
popdens	4.082916
barren	0.018703736
water	-0.97532403
nature	-0.97348666
agric	-0.9760582

Predicted Urban for 2016

# Southeastern USA



# Predictions for North Carolina



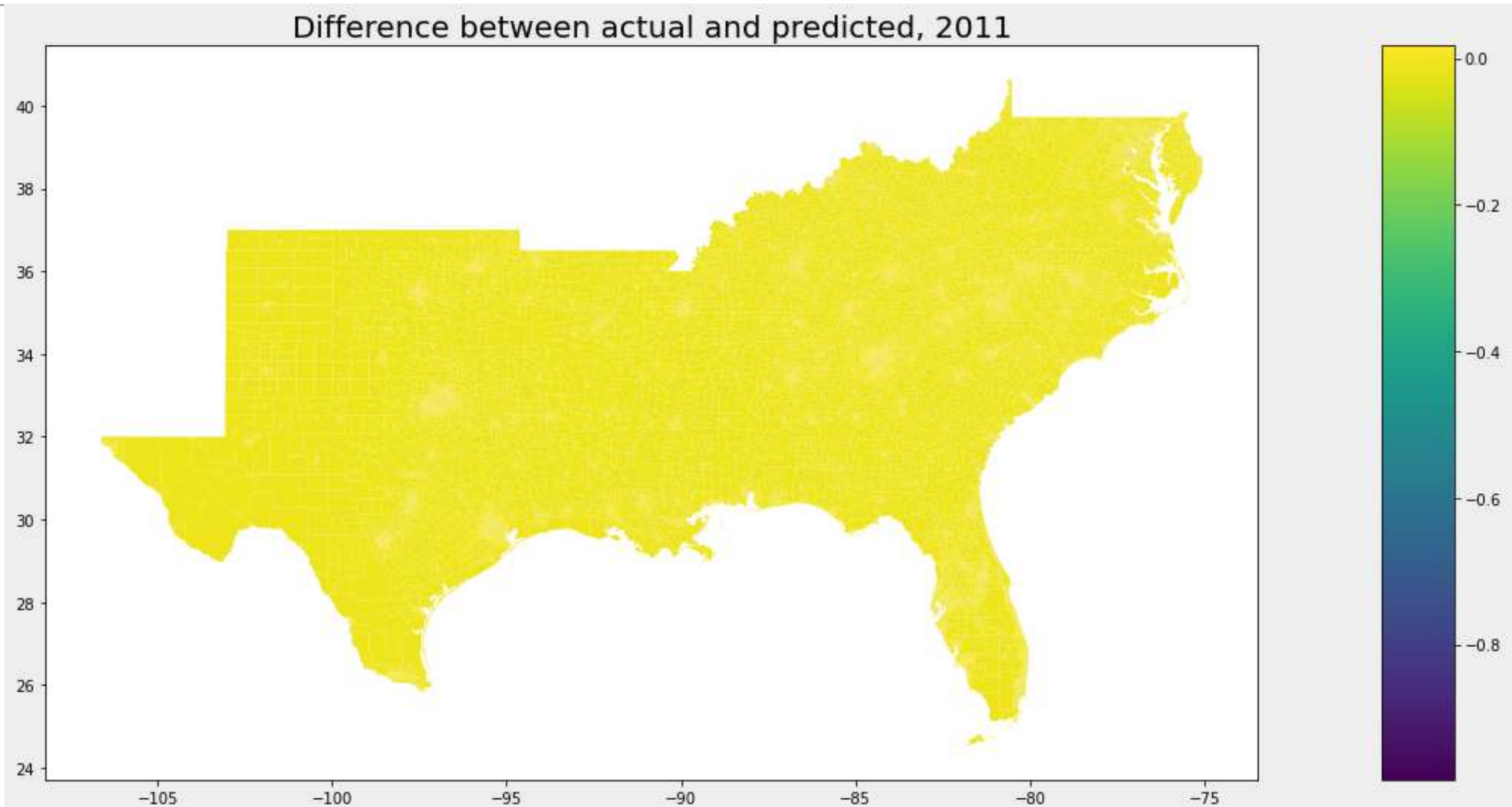
# Performance Evaluation

---

- Manually perform prediction
  - Model overestimates percent-urban (Y) by 0.5% for a given census tract
  - 0.5% of a census tract (~2,436 acres) is approximately 121 acres
- R-Squared
  - Calculated in Spark using distributed DataFrames
  - Result: 0.98 → Good enough



# Visually inspect performance (less is best)



# Possible Improvements

---

- Use RasterFrames on the cluster for raster processing
  - <https://rasterframes.io/getting-started.html>
  - Removes the ArcGIS Pro dependency in the data pipeline
- Use 'census' or 'CensusData' packages in Python
  - Removes the R dependency in the data pipeline

A solid green vertical bar is positioned on the far left side of the slide.

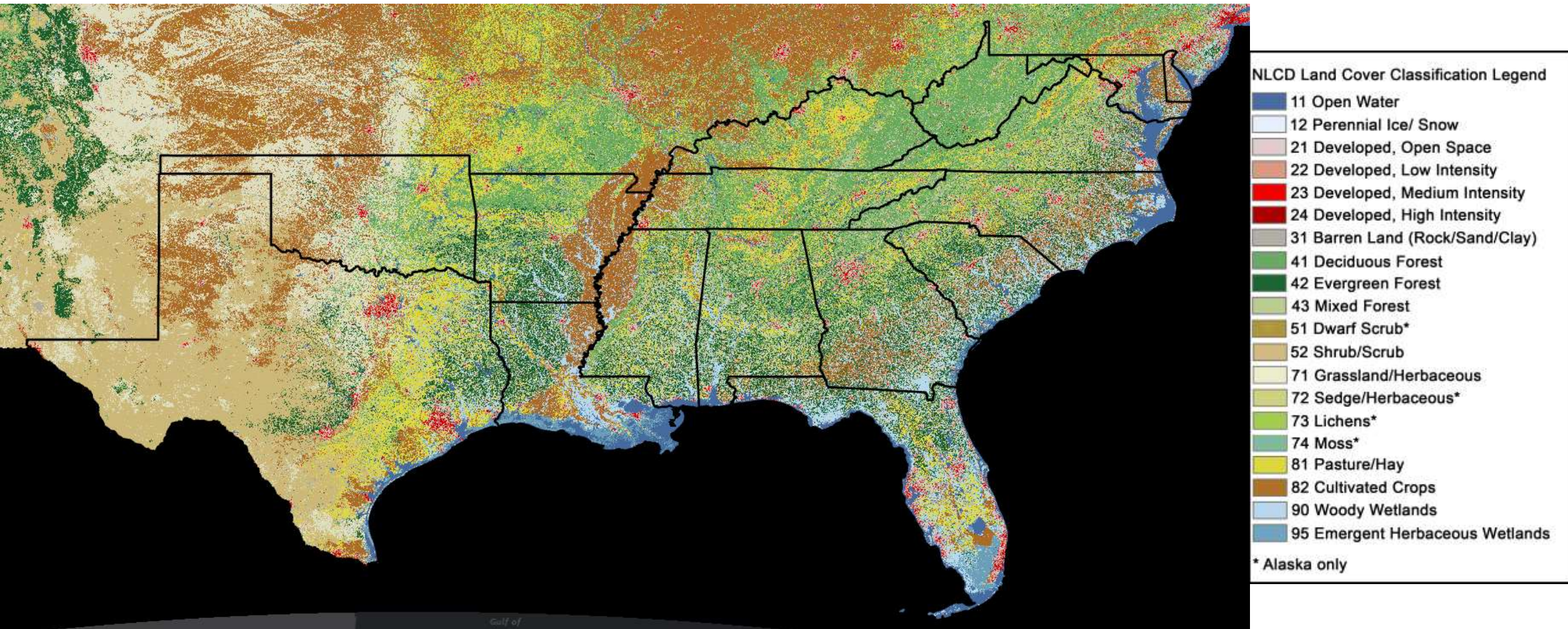
# End.

---

Next: Screenshots



# National Land Cover Database





# Roadways

---

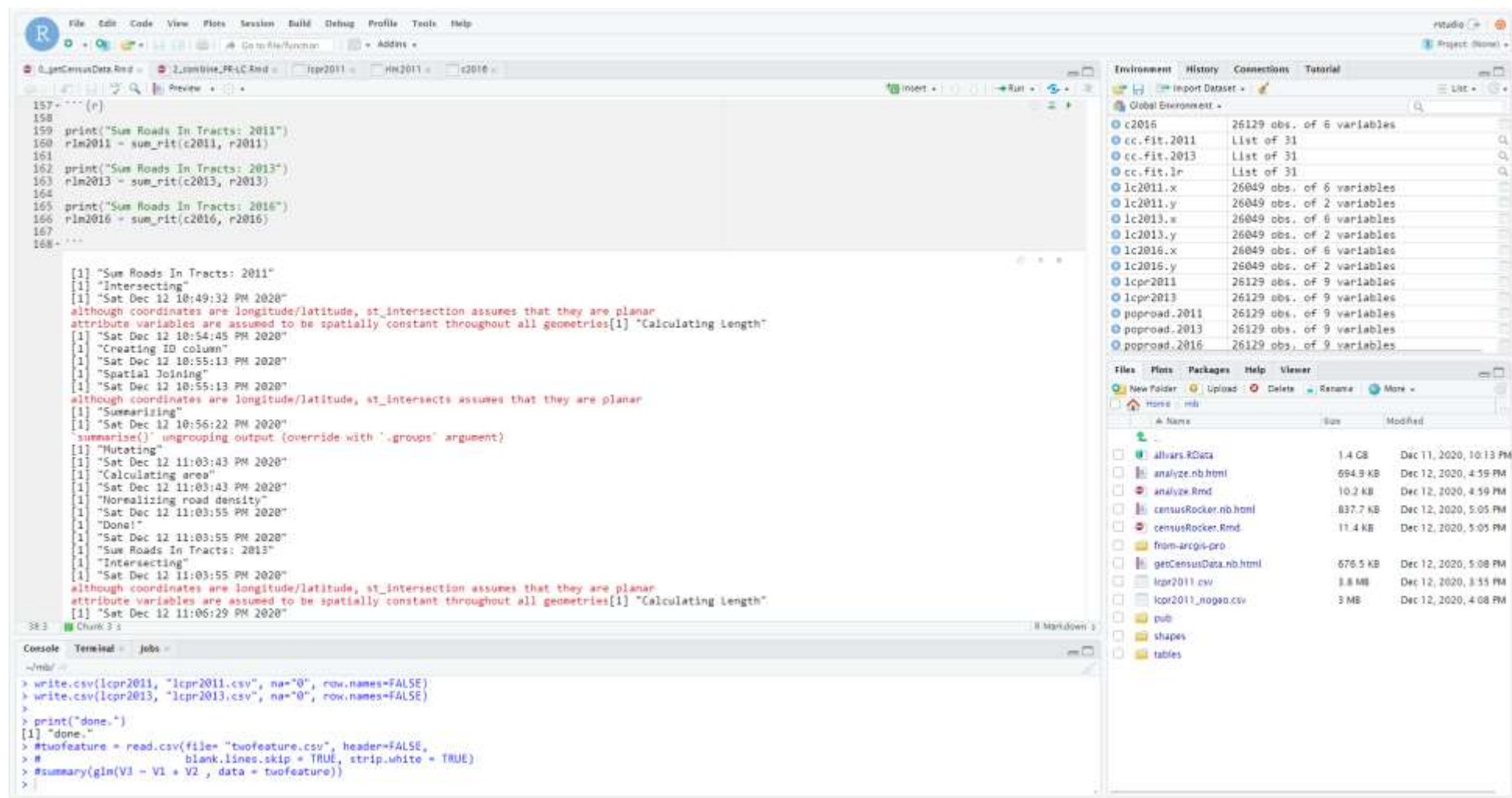


# Census Tracts

---



# R Studio



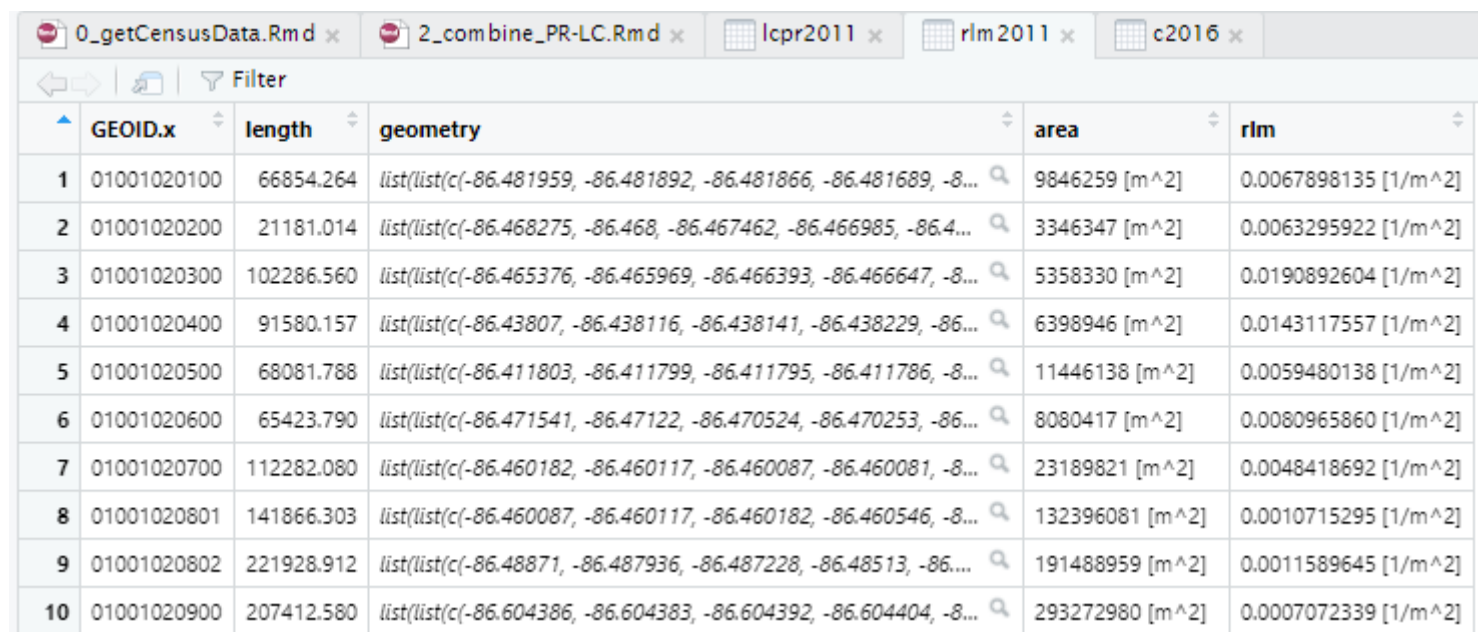


# Census tract data

0_getCensusData.Rmd x 2_combine_PR-LC.Rmd x lcpr2011 x rlm2011 x c2016 x						
Filter						
	GEOID	NAME	variable	estimate	moe	geometry
1	01099076200	Census Tract 762, Monroe County, Alabama	B01003_001	1637	283	list(list(c(-87.784378, -87.781681, -87.780076, -87.771374, -8...
2	01101002100	Census Tract 21, Montgomery County, Alabama	B01003_001	4027	384	list(list(c(-86.299615, -86.294088, -86.290362, -86.288723, -8...
3	01101005302	Census Tract 53.02, Montgomery County, Alabama	B01003_001	2178	272	list(list(c(-86.23964, -86.237495, -86.23762, -86.239558, -86...
4	01101005606	Census Tract 56.06, Montgomery County, Alabama	B01003_001	4203	699	list(list(c(-86.233767, -86.232232, -86.23161, -86.22884, -86...
5	01103000300	Census Tract 3, Morgan County, Alabama	B01003_001	2850	366	list(list(c(-86.984562, -86.976697, -86.972917, -86.967527, -8...
6	01103005303	Census Tract 53.03, Morgan County, Alabama	B01003_001	3871	355	list(list(c(-86.95923, -86.959222, -86.956708, -86.954739, -86...
7	01103005500	Census Tract 55, Morgan County, Alabama	B01003_001	5356	483	list(list(c(-87.028764, -87.027291, -87.025354, -87.023192, -8...
8	01105686800	Census Tract 6868, Perry County, Alabama	B01003_001	1104	279	list(list(c(-87.273701, -87.270902, -87.269211, -87.26494, -87...
9	01111000100	Census Tract 1, Randolph County, Alabama	B01003_001	3325	372	list(list(c(-85.648139, -85.646238, -85.64599, -85.644217, -85...
10	01111000300	Census Tract 3, Randolph County, Alabama	B01003_001	3934	386	list(list(c(-85.651041, -85.651138, -85.650466, -85.641435, -8...



# Road density dataset



	GEOID.x	length	geometry	area	rlm
1	01001020100	66854.264	list(list(c(-86.481959, -86.481892, -86.481866, -86.481689, -8...	9846259 [m^2]	0.0067898135 [1/m^2]
2	01001020200	21181.014	list(list(c(-86.468275, -86.468, -86.467462, -86.466985, -86.4...	3346347 [m^2]	0.0063295922 [1/m^2]
3	01001020300	102286.560	list(list(c(-86.465376, -86.465969, -86.466393, -86.466647, -8...	5358330 [m^2]	0.0190892604 [1/m^2]
4	01001020400	91580.157	list(list(c(-86.43807, -86.438116, -86.438141, -86.438229, -86...	6398946 [m^2]	0.0143117557 [1/m^2]
5	01001020500	68081.788	list(list(c(-86.411803, -86.411799, -86.411795, -86.411786, -8...	11446138 [m^2]	0.0059480138 [1/m^2]
6	01001020600	65423.790	list(list(c(-86.471541, -86.47122, -86.470524, -86.470253, -86...	8080417 [m^2]	0.0080965860 [1/m^2]
7	01001020700	112282.080	list(list(c(-86.460182, -86.460117, -86.460087, -86.460081, -8...	23189821 [m^2]	0.0048418692 [1/m^2]
8	01001020801	141866.303	list(list(c(-86.460087, -86.460117, -86.460182, -86.460546, -8...	132396081 [m^2]	0.0010715295 [1/m^2]
9	01001020802	221928.912	list(list(c(-86.48871, -86.487936, -86.487228, -86.48513, -86...	191488959 [m^2]	0.0011589645 [1/m^2]
10	01001020900	207412.580	list(list(c(-86.604386, -86.604383, -86.604392, -86.604404, -8...	293272980 [m^2]	0.0007072339 [1/m^2]

$\text{length (meters)} / \text{area (m}^2\text{)} = \text{rlm (meters of road per square meter of census tract)}$

# 0\_getcensusdata

---

```
169
170
171 # Write shapefiles to disk
172 ## Census Tracts w/ Pop and Road Lines
173 ```{r}
174 st_write(r2011, "shapes/r2011.shp", delete_layer = TRUE)
175 st_write(r2013, "shapes/r2013.shp", delete_layer = TRUE)
176 st_write(r2016, "shapes/r2016.shp", delete_layer = TRUE)
177
178 st_write(c2011, "shapes/c2011.shp", delete_layer = TRUE)
179 st_write(c2013, "shapes/c2013.shp", delete_layer = TRUE)
180 st_write(c2016, "shapes/c2016.shp", delete_layer = TRUE)
181 ^```
```

```
Writing layer `r2011' to data source `shapes/r2011.shp' using driver `ESRI Shapefile'
Writing 159266 features with 5 fields and geometry type Multi Line String.
Writing layer `r2013' to data source `shapes/r2013.shp' using driver `ESRI Shapefile'
Writing 141574 features with 4 fields and geometry type Unknown (any).
Writing layer `r2016' to data source `shapes/r2016.shp' using driver `ESRI Shapefile'
Writing 141041 features with 5 fields and geometry type Unknown (any).
Writing layer `c2011' to data source `shapes/c2011.shp' using driver `ESRI Shapefile'
Writing 26129 features with 5 fields and geometry type Multi Polygon.
Writing layer `c2013' to data source `shapes/c2013.shp' using driver `ESRI Shapefile'
Writing 26129 features with 5 fields and geometry type Multi Polygon.
Writing layer `c2016' to data source `shapes/c2016.shp' using driver `ESRI Shapefile'
Writing 26129 features with 5 fields and geometry type Multi Polygon.
```

# GCP: Dataproc

## Cluster configuration page

The screenshot shows the Google Cloud Platform interface for a Dataproc cluster named 'cluster-95c6'. The top navigation bar includes the Google Cloud Platform logo, the user 'gphotodl', and a search bar. The left sidebar contains navigation icons for various GCP services. The main content area has tabs for MONITORING, JOBS, VM INSTANCES, CONFIGURATION (selected), and WEB INTERFACES. An 'EDIT' button is visible at the top of the CONFIGURATION tab. The configuration details are as follows:

Property	Value
Region	us-east4
Zone	us-east4-c
Autoscaling	Off
Scheduled deletion	The cluster will be deleted at December 23, 2020 at 9:53:54 PM UTC-5
Enhanced flexibility mode	Off
Master node	Standard (1 master, N workers)
Machine type	n1-standard-2
Number of GPUs	0
Primary disk type	pd-standard
Primary disk size	100GB
Local SSDs	0
Worker nodes	3
Machine type	n1-standard-2
Number of GPUs	0
Primary disk type	pd-standard
Primary disk size	100GB
Local SSDs	0
Secondary worker nodes	0
Cloud Storage staging bucket	<a href="#">frzrbrn-bk1</a>
Network	default
Network tags	None
Internal IP only	No
Image version	1.5.24-ubuntu18
Created	Dec 12, 2020, 9:53:54 PM
Optional components	JUPYTER ZEPPELIN ANACONDA
Properties	<a href="#">SHOW PROPERTIES</a>
Advanced security	Disabled
Labels	goog-datap... : cluster-95... <a href="#">▼</a>
Encryption type	Google-managed key

# GCP: Dataproc

Web interfaces available through  
'Component gateway'

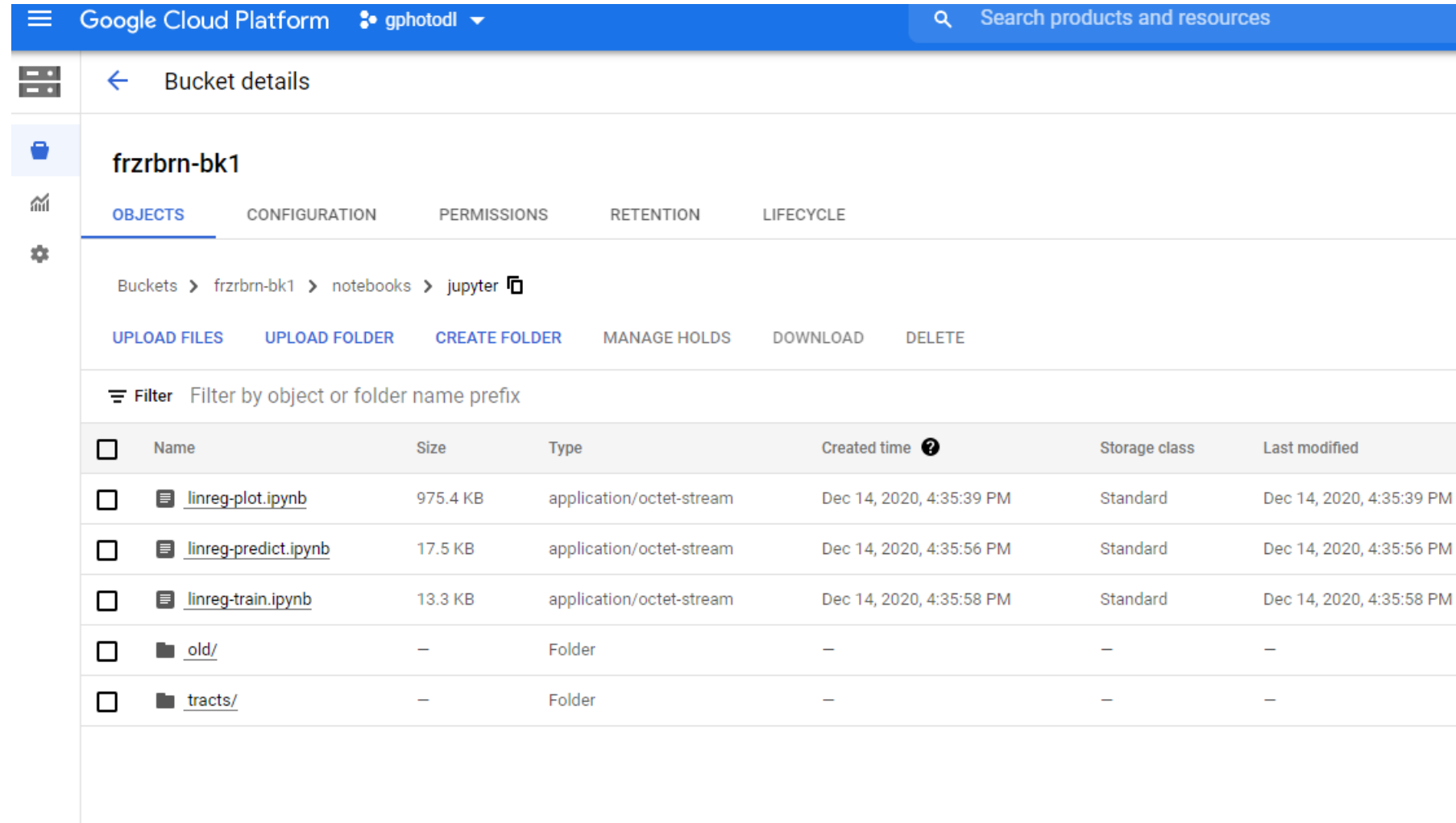
The screenshot shows the Google Cloud Platform interface for a Dataproc cluster named 'cluster-95c6'. The top navigation bar includes the Google Cloud logo, the text 'Google Cloud Platform', and a user profile icon labeled 'gphotodl'. Below the navigation bar, the cluster name 'cluster-95c6' is displayed, along with action buttons: 'SUBMIT JOB', 'REFRESH', 'DELETE', and 'VIEW LOGS'. A sidebar on the left contains icons for various GCP services. The main content area features a table with cluster details:

Name	cluster-95c6
Cluster UUID	f671aeef-38c6-4bce-ab14-ca1ecd109c0e
Type	Dataproc Cluster
Status	Running

Below the table, there are tabs for 'MONITORING', 'JOBS', 'VM INSTANCES', 'CONFIGURATION', and 'WEB INTERFACES'. The 'WEB INTERFACES' tab is selected, showing a section for 'SSH tunnel' with a link to 'Create an SSH tunnel to connect to a web interface'. Below this, the 'Component gateway' section provides access to various web interfaces, each with a link and an external icon:

- YARN ResourceManager
- MapReduce Job History
- YARN Application Timeline
- Spark History Server
- HDFS NameNode
- Tez
- Jupyter
- JupyterLab
- Zeppelin
- Equivalent REST

# GCP: Bucket access



Google Cloud Platform gphotodl Search products and resources

Bucket details






frzrbrn-bk1

OBJECTS CONFIGURATION PERMISSIONS RETENTION LIFECYCLE




Buckets > frzrbrn-bk1 > notebooks > jupyter










UPLOAD FILES UPLOAD FOLDER CREATE FOLDER MANAGE HOLDS DOWNLOAD DELETE




Filter Filter by object or folder name prefix





<input type="checkbox"/>	Name	Size	Type	Created time ?	Storage class	Last modified
<input type="checkbox"/>	 <a href="#">linreg-plot.ipynb</a>	975.4 KB	application/octet-stream	Dec 14, 2020, 4:35:39 PM	Standard	Dec 14, 2020, 4:35:39 PM
<input type="checkbox"/>	 <a href="#">linreg-predict.ipynb</a>	17.5 KB	application/octet-stream	Dec 14, 2020, 4:35:56 PM	Standard	Dec 14, 2020, 4:35:56 PM
<input type="checkbox"/>	 <a href="#">linreg-train.ipynb</a>	13.3 KB	application/octet-stream	Dec 14, 2020, 4:35:58 PM	Standard	Dec 14, 2020, 4:35:58 PM
<input type="checkbox"/>	 <a href="#">old/</a>	—	Folder	—	—	—
<input type="checkbox"/>	 <a href="#">tracts/</a>	—	Folder	—	—	—

# GCP: Cluster Virtual Machines

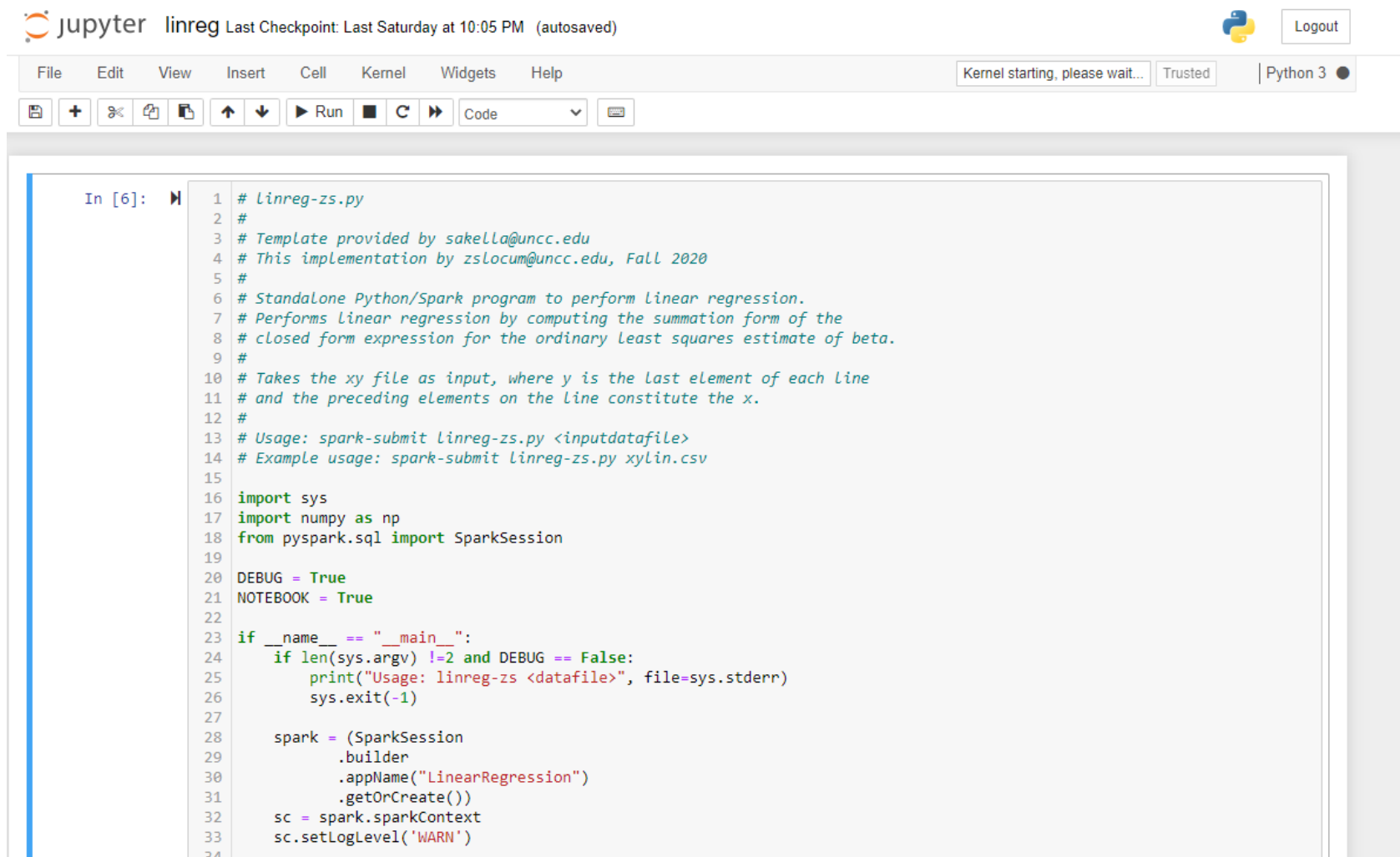
 Google Cloud Platform  gphotodl  Search products and resources

 VM instances  CREATE INSTANCE  IMPORT VM  REFRESH  START / RESUME  STOP  SUSPEND  RESET  DELETE

 Filter VM instances  Columns 

<input checked="" type="checkbox"/> Name ^	Zone	Recommendation	In use by	Internal IP	External IP	Connect
<input checked="" type="checkbox"/>  cluster-95c6-m	us-east4-c.			10.150.0.12 (nic0)	None	SSH ▾ ⋮
<input checked="" type="checkbox"/>  cluster-95c6-w-0	us-east4-c.			10.150.0.11 (nic0)	None	SSH ▾ ⋮
<input checked="" type="checkbox"/>  cluster-95c6-w-1	us-east4-c.			10.150.0.13 (nic0)	None	SSH ▾ ⋮
<input checked="" type="checkbox"/>  cluster-95c6-w-2	us-east4-c.			10.150.0.14 (nic0)	None	SSH ▾ ⋮

# Local Jupyter Notebook development



The screenshot displays a Jupyter Notebook interface. At the top, the header shows 'jupyter linreg' and 'Last Checkpoint: Last Saturday at 10:05 PM (autosaved)'. On the right, there is a 'Logout' button and a Python logo. Below the header is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. To the right of the menu bar, there are buttons for 'Kernel starting, please wait...', 'Trusted', and 'Python 3'. Below the menu bar is a toolbar with icons for saving, adding cells, zooming, copying, pasting, undo, redo, and running code. The main area of the notebook shows a code cell with the following Python code:

```
In [6]: 1 # linreg-zs.py
2 #
3 # Template provided by sakella@uncc.edu
4 # This implementation by zslocum@uncc.edu, Fall 2020
5 #
6 # Standalone Python/Spark program to perform linear regression.
7 # Performs linear regression by computing the summation form of the
8 # closed form expression for the ordinary Least squares estimate of beta.
9 #
10 # Takes the xy file as input, where y is the last element of each line
11 # and the preceding elements on the line constitute the x.
12 #
13 # Usage: spark-submit linreg-zs.py <inputdatafile>
14 # Example usage: spark-submit linreg-zs.py xylin.csv
15
16 import sys
17 import numpy as np
18 from pyspark.sql import SparkSession
19
20 DEBUG = True
21 NOTEBOOK = True
22
23 if __name__ == "__main__":
24     if len(sys.argv) != 2 and DEBUG == False:
25         print("Usage: linreg-zs <datafile>", file=sys.stderr)
26         sys.exit(-1)
27
28     spark = (SparkSession
29             .builder
30             .appName("LinearRegression")
31             .getOrCreate())
32     sc = spark.sparkContext
33     sc.setLogLevel('WARN')
34
```

# Run Jupyter Notebook Server with pyspark

---

```
zachery@ubu20lake:~/cc-jnb$ cat run-spark-nb.sh
#!/bin/bash

export PYSPARK_DRIVER_PYTHON=jupyter
export PYSPARK_DRIVER_PYTHON_OPTS='notebook --config /opt/hadoop-3.3.0/pyspark-jnb-config.py'

nohup pyspark &
```



# GCP Billing page

