CIS 8392
Topics in Big Data Analytics

#R and GCP

Yu-Kai Lin

Agenda

- googleCloudStorageR
- bigrquery
- googleLanguageR
- RoogleVision
- googleComputeEngineR

Prerequisites

IMPORTANT: Do not run all the `install.packages` commands at once. Run it line by line. You may be asked whether to install/update some additional packages. If so, choose **none**.

```r
install.packages("png")
install.packages("wk")
install.packages("googleCloudStorageR")
install.packages("googleLanguageR")
install.packages("devtools")
install.packages("bigrquery")
devtools::install_github("cloudyr/RoogleVision")
devtools::install_github("cloudyr/googleComputeEngineR")

library(tidyverse)
library(bigrquery)
library(googleCloudStorageR)
library(googleLanguageR)
library(RoogleVision)
library(googleComputeEngineR)
```

Create a credential

https://console.developers.google.com/apis/credentials/consent

- **Application name**: cis8392
- **Authorized domains**: gsu.edu
- information about links: http://gsu.edu
- Click the **Save** button at the bottom of the page

https://console.developers.google.com/apis/credentials

- In the **Create credentials** dropdown box, select **OAuth client ID**
- **Application type**: Web app
- **Name**: any name is fine
- Click the **Create** button at the bottom of the page

https://console.developers.google.com/apis/credentials

- You should see the client showing up on the page
- Click the download icon (⤓) on the right
- rename the file to **gcp-auth.json** and move it to your R working directory

Create a service account key

Go to https://console.cloud.google.com/iam-admin/serviceaccounts

- If prompted, select your project

Then, under the action column

- click the More button (**vertial dot-dot-dot sign**)
- Select **Create key**
- Key type: **JSON**

Your browser should then automatically download a JSON file.

- Rename the file to `gcp-service-account-key.json`
- Move the file to your R working directory

Enable APIs

Enable the following GCP APIs for your project:

API	URL
Cloud Vision	https://console.developers.google.com/apis/api/vision.googleapis.com
Cloud Natural Language	https://console.developers.google.com/apis/api/language.googleapis.com
Cloud Speech-to-Text	https://console.cloud.google.com/marketplace/product/google/speech.googleapis.com
Cloud Text-to-Speech	https://console.developers.google.com/apis/api/texttospeech.googleapis.com
Cloud Translation	https://console.developers.google.com/apis/api/translate.googleapis.com
Compute Engine	https://console.developers.google.com/apis/api/compute.googleapis.com

googleCloudStorageR

```r
#install.packages("googleCloudStorageR")
library(googleCloudStorageR)

Sys.setenv("GCS_AUTH_FILE" = "gcp-service-account-key.json")
options("googleAuthR.scopes.selected" =
        "https://www.googleapis.com/auth/cloud-platform")
proj_id = googleAuthR::gar_set_client("gcp-auth.json")
gcs_auth("gcp-service-account-key.json")

buckets <- gcs_list_buckets(proj_id)
buckets
```

```
##                              name storageClass location             updated
## 1                         cis8392     STANDARD US-EAST1 2019-12-02 15:28:16
## 2 cis8392-20210422-yukai-bucket     STANDARD US-EAST1 2021-04-22 22:56:42
## 3                  cis8392-bucket     STANDARD US-EAST1 2019-12-02 15:28:06
## 4                cis8392-bucket-2     STANDARD US-EAST1 2021-04-22 22:31:44
## 5             cis8392-patent-test     STANDARD       US 2020-10-24 19:44:52
## 6            cis8392-yklin-bucket     STANDARD US-EAST1 2020-12-02 23:44:00
```

Create a bucket:

Remember that the bucket name has to be globally unique on GCP. You cannot use `cis8392-bucket-tmp` if I have already taken this bucket name.

```r
gcs_create_bucket(name="cis8392-bucket-tmp", projectId = proj_id)
```

```
## ==Google Cloud Storage Bucket==
## Bucket:          cis8392-bucket-tmp
## Project Number:  138962636551
## Location:        US
## Class:           MULTI_REGIONAL
## Created:         2021-09-08 18:51:03
## Updated:         2021-09-08 18:51:03
## Versioning:      FALSE
## Meta-generation: 1
## eTag:            CAE=
```

```r
buckets <- gcs_list_buckets(proj_id)
buckets
```

```
##                            name   storageClass location             updated
## 1                       cis8392       STANDARD US-EAST1 2019-12-02 15:28:16
## 2 cis8392-20210422-yukai-bucket       STANDARD US-EAST1 2021-04-22 22:56:42
## 3                cis8392-bucket       STANDARD US-EAST1 2019-12-02 15:28:06
## 4              cis8392-bucket-2       STANDARD US-EAST1 2021-04-22 22:31:44
```

Delete a bucket:

```
gcs_delete_bucket("cis8392-bucket-tmp")
```

```
## [1] TRUE
```

```
buckets <- gcs_list_buckets(proj_id)
buckets
```

```
##                            name storageClass location             updated
## 1                       cis8392     STANDARD US-EAST1 2019-12-02 15:28:16
## 2 cis8392-20210422-yukai-bucket     STANDARD US-EAST1 2021-04-22 22:56:42
## 3                cis8392-bucket     STANDARD US-EAST1 2019-12-02 15:28:06
## 4              cis8392-bucket-2     STANDARD US-EAST1 2021-04-22 22:31:44
## 5           cis8392-patent-test     STANDARD       US 2020-10-24 19:44:52
## 6          cis8392-yklin-bucket     STANDARD US-EAST1 2020-12-02 23:44:00
```

Set a default bucket:

```r
gcs_global_bucket("cis8392-bucket")
gcs_get_global_bucket()
```

```
## [1] "cis8392-bucket"
```

This is useful when you plan to use the `googleCloudStorageR` package to access data from the same bucket multiple times in the current R session.

Upload an object:

If the data source is a file (make sure `HousePrices.csv` is in your working directory or you can download it here):

```
gcs_upload("data/HousePrices.csv")
```

```
## ==Google Cloud Storage Object==
## Name:                 HousePrices.csv
## Type:                 text/csv
## Size:                 27.9 Kb
## Media URL:            https://www.googleapis.com/download/storage/v1/b/cis8392-buc
## Download URL:         https://storage.cloud.google.com/cis8392-bucket/HousePrices.
## Public Download URL:  https://storage.googleapis.com/cis8392-bucket/HousePrices.cs
## Bucket:               cis8392-bucket
## ID:                   cis8392-bucket/HousePrices.csv/1631127064348065
## MD5 Hash:             Px8m37OvkYDVSf8/Ffkytg==
## Class:                STANDARD
## Created:              2021-09-08 18:51:04
## Updated:              2021-09-08 18:51:04
## Generation:           1631127064348065
## Meta Generation:      1
## eTag:                 CKGLhOqF8PICEAE=
## crc32c:               QJmUdg==
```

Upload an object:

If the data source is an R object:

```
gcs_upload(
  list(a = 1, b = 3, c = list(d = 2, e = 5)),
  name="my_list.json")
```

```
objects <- gcs_list_objects()
objects$name
```

```
## [1] "HousePrices.csv"            "just-a-folder/kitten3.png"
## [3] "kitten.png"                 "mtcars_test1.csv"
## [5] "my_list.json"
```

Get objects from a bucket:

```r
objects <- gcs_list_objects(bucket="cis8392-bucket") # manually specify bucket
objects <- gcs_list_objects() # from the default bucket
class(objects)
```

```
## [1] "data.frame"
```

```r
objects$name
```

```
## [1] "HousePrices.csv"             "just-a-folder/kitten3.png"
## [3] "kitten.png"                  "mtcars_test1.csv"
## [5] "my_list.json"
```

```r
objects %>% as_tibble() # pretty print
```

```
## # A tibble: 5 x 3
##   name                      size     updated
##   <chr>                     <chr>    <dttm>
## 1 HousePrices.csv           27.9 Kb  2021-09-08 18:51:04
## 2 just-a-folder/kitten3.png 129.8 Kb 2021-04-22 22:37:01
## 3 kitten.png                129.8 Kb 2021-04-22 22:40:01
## 4 mtcars_test1.csv          1.2 Kb   2020-12-02 16:37:17
## 5 my_list.json              41 bytes 2021-04-23 00:16:18
```

Download and save an object to R:

- The download type is guessed and converted into an appropriate R object
- **WARNING:** The object could be very large. Make sure you don't run out of RAM

```
bucket="cis8392-bucket"
kitten <- gcs_get_object(object_name = "kitten.png")
str(kitten)
```

```
##  num [1:251, 1:384, 1:3] 0.0549 0.0667 0.051 0.0667 0.0745 ...
```

```
plot(as.raster(kitten))
```

```r
df <- gcs_get_object(object_name = "HousePrices.csv")
df
```

```
## # A tibble: 546 x 12
##    price lotsize bedrooms bathrooms stories driveway recreation fullbase gasheat
##    <dbl>   <dbl>    <dbl>     <dbl>   <dbl> <chr>    <chr>      <chr>    <chr>
##  1 42000    5850        3         1       2 yes      no         yes      no
##  2 38500    4000        2         1       1 yes      no         no       no
##  3 49500    3060        3         1       1 yes      no         no       no
##  4 60500    6650        3         1       2 yes      yes        no       no
##  5 61000    6360        2         1       1 yes      no         no       no
##  6 66000    4160        3         1       1 yes      yes        yes      no
##  7 66000    3880        3         2       2 yes      no         yes      no
##  8 69000    4160        3         1       3 yes      no         no       no
##  9 83800    4800        3         1       1 yes      yes        yes      no
## 10 88500    5500        3         2       4 yes      yes        no       no
## # ... with 536 more rows, and 3 more variables: aircon <chr>, garage <dbl>,
## #   prefer <chr>
```

If you want to get the metadata of an object rather than the object itself:

```r
gcs_get_object("HousePrices.csv", meta = TRUE)
```

```
## ==Google Cloud Storage Object==
## Name:                HousePrices.csv
## Type:                text/csv
## Size:                27.9 Kb
## Media URL:           https://www.googleapis.com/download/storage/v1/b/cis8392-buc
## Download URL:        https://storage.cloud.google.com/cis8392-bucket/HousePrices.
## Public Download URL: https://storage.googleapis.com/cis8392-bucket/HousePrices.cs
## Bucket:              cis8392-bucket
## ID:                  cis8392-bucket/HousePrices.csv/1631127064348065
## MD5 Hash:            Px8m37OvkYDVSf8/Ffkytg==
## Class:               STANDARD
## Created:             2021-09-08 18:51:04
## Updated:             2021-09-08 18:51:04
## Generation:          1631127064348065
## Meta Generation:     1
## eTag:                CKGLhOqF8PICEAE=
## crc32c:              QJmUdg==
```

Update user access to objects:

Update access of object to READER for all users (that is, making it publicly accesible):

```
gcs_update_object_acl("HousePrices.csv", entity_type = "allUsers")
```

```
## [1] TRUE
```

Update access of object for user gsu.cis.8392@gmail.com to OWNER:

```
gcs_update_object_acl("HousePrices.csv",
                      entity = "gsu.cis.8392@gmail.com",
                      role = "OWNER")
```

```
## [1] TRUE
```

```
acl <- gcs_get_object_acl("HousePrices.csv",
                          entity = "gsu.cis.8392@gmail.com")
acl$role
```

```
## [1] "OWNER"
```

Create a download link for an object:

```
download_url <- gcs_download_url("HousePrices.csv")
download_url
```

```
## [1] "https://storage.cloud.google.com/cis8392-bucket/HousePrices.csv"
```

Delete an object:

```
gcs_delete_object("HousePrices.csv")
```

```
## [1] TRUE
```

```
objects <- gcs_list_objects()
objects$name
```

```
## [1] "just-a-folder/kitten3.png" "kitten.png"
## [3] "mtcars_test1.csv"          "my_list.json"
```

bigrquery

```r
library(bigrquery)
library(DBI)

bq_auth(path = "gcp-service-account-key.json")
```

Create a BigQuery dataset

The `bigrquery` package does not allow you to create a BigQuery dataset in R. So we need to create a dataset in the project manually.

Go to https://console.cloud.google.com/bigquery

1. Click your project name from the list on the left (my project name is `cis8392-260815`)
2. Click **CREATE DATASET** on the right (below the Query editor)
    - Dataset ID: **bq_exercise**
    - Leave all else with their default values
    - Click the **Create dataset** button at the bottom

Adding some data into the dataset

We are going to use COVID data from Johns Hopkins University

The data is available on Google BigQuery.

- Go to https://console.cloud.google.com/bigquery
- One the left, in the **Type to search** box, enter `covid19_jhu_csse`
- It should show **Found 0 results.** Then click `Broaden search to all projects`.
- There are several tables under `covid19_jhu_csse`

Let's copy the following tables to your own **bq_exercise** dataset: `confirmed_cases`, `deaths`, `recovered_cases`

- To copy a table, click each of the tables from the list on the left under `covid19_jhu_csse`
- You should see the **COPY TABLE** option on the right, which allows you to copy the table to your own project.

List tables within your project:

```r
projects <- bq_projects()
projects
```

```
## [1] "cis8392-260815"
```

```r
con <- dbConnect(
  bigrquery::bigquery(),
  project = "cis8392-260815", # use your own project ID!
  dataset = "bq_exercise",
  billing = "cis8392-260815"  # use your own project ID!
)
con
```

```
## <BigQueryConnection>
##   Dataset: cis8392-260815.bq_exercise
##   Billing: cis8392-260815
```

```r
dbListTables(con)
```

```
## [1] "confirmed_cases" "deaths"          "recovered_cases"
```

Query data:

```r
# you can use SQL to query the table
sql = "select * from confirmed_cases"
confirmed_df = dbGetQuery(con, sql, n = 1000)

# if you just want to get the entire table
confirmed_df2 <- bq_table_download(
  "cis8392-260815.bq_exercise.confirmed_cases",
  max_results = 1000)

# you can also get data directly from the source (covid19_jhu_csse)
confirmed_df3 <- bq_table_download(
  "bigquery-public-data.covid19_jhu_csse.confirmed_cases",
  max_results = 1000)
```

Your turn

Use `bigrquery` (and `tidyverse`) and the `covid19_jhu_csse` data to find out the following information and use ggplot to visualize each of them:

1. Top 5 regions based on the latest number of confirmed cases

2. Monthly trend of COVID recovered cases in the US

googleLanguageR

Input text: https://dl.dropboxusercontent.com/s/ybbdyvqp1jlgqcl/gsu.txt

```r
library(googleLanguageR)
gl_auth("gcp-service-account-key.json")

url = "https://dl.dropboxusercontent.com/s/ybbdyvqp1jlgqcl/gsu.txt"
text <- read_file(url)

nlp_result <- gl_nlp(text)
str(nlp_result, max.level = 2)
```

```
## List of 7
##  $ sentences        :List of 1
##   ..$ :'data.frame':    3 obs. of  4 variables:
##  $ tokens           :List of 1
##   ..$ :'data.frame':    84 obs. of  17 variables:
##  $ entities         :List of 1
##   ..$ : tibble [32 x 11] (S3: tbl_df/tbl/data.frame)
##  $ language         : chr "en"
##  $ text             : chr "Georgia State University (Georgia State, State, or GSU
##  $ documentSentiment: tibble [1 x 2] (S3: tbl_df/tbl/data.frame)
##  $ classifyText     :List of 1
##   ..$ : tibble [1 x 2] (S3: tbl_df/tbl/data.frame)
```

Sentences:

```
glimpse(nlp_result$sentences[[1]]) #View(nlp_result$sentences[[1]])
```

```
## Rows: 3
## Columns: 4
## $ content     <chr> "Georgia State University (Georgia State, State, or GSU) i~
## $ beginOffset <int> 0, 109, 202
## $ magnitude   <dbl> 0.0, 0.0, 0.5
## $ score       <dbl> 0.0, 0.0, 0.5
```

```
nlp_result$sentences[[1]]
```

```
##
## 1
## 2
## 3 It is also the largest institution of higher education based in Georgia and is
##   beginOffset magnitude score
## 1           0       0.0   0.0
## 2         109       0.0   0.0
## 3         202       0.5   0.5
```

Tokens:

```
glimpse(nlp_result$tokens[[1]]) #View(nlp_result$tokens[[1]])
```

```
## Rows: 84
## Columns: 17
## $ content        <chr> "Georgia", "State", "University", "(", "Georgia", "Stat~
## $ beginOffset    <int> 0, 8, 14, 25, 26, 34, 39, 41, 46, 48, 51, 54, 56, 59, 6~
## $ tag            <chr> "NOUN", "NOUN", "NOUN", "PUNCT", "NOUN", "NOUN", "PUNCT~
## $ aspect         <chr> "ASPECT_UNKNOWN", "ASPECT_UNKNOWN", "ASPECT_UNKNOWN", "~
## $ case           <chr> "CASE_UNKNOWN", "CASE_UNKNOWN", "CASE_UNKNOWN", "CASE_U~
## $ form           <chr> "FORM_UNKNOWN", "FORM_UNKNOWN", "FORM_UNKNOWN", "FORM_U~
## $ gender         <chr> "GENDER_UNKNOWN", "GENDER_UNKNOWN", "GENDER_UNKNOWN", "~
## $ mood           <chr> "MOOD_UNKNOWN", "MOOD_UNKNOWN", "MOOD_UNKNOWN", "MOOD_U~
## $ number         <chr> "SINGULAR", "SINGULAR", "SINGULAR", "NUMBER_UNKNOWN", "~
## $ person         <chr> "PERSON_UNKNOWN", "PERSON_UNKNOWN", "PERSON_UNKNOWN", "~
## $ proper         <chr> "PROPER", "PROPER", "PROPER", "PROPER_UNKNOWN", "PROPER~
## $ reciprocity    <chr> "RECIPROCITY_UNKNOWN", "RECIPROCITY_UNKNOWN", "RECIPROC~
## $ tense          <chr> "TENSE_UNKNOWN", "TENSE_UNKNOWN", "TENSE_UNKNOWN", "TEN~
## $ voice          <chr> "VOICE_UNKNOWN", "VOICE_UNKNOWN", "VOICE_UNKNOWN", "VOI~
## $ headTokenIndex <int> 2, 2, 12, 2, 5, 2, 5, 5, 5, 5, 5, 2, 12, 16, 16, 16, 12~
## $ label          <chr> "NN", "NN", "NSUBJ", "P", "NN", "APPOS", "P", "CONJ", "~
## $ value          <chr> "Georgia", "State", "University", "(", "Georgia", "Stat~
```

Entities

```
nlp_result$entities[[1]]
```

```
## # A tibble: 32 x 11
##    name   type   salience wikipedia_url mid   year  value magnitude score
##    <chr>  <chr>     <dbl> <chr>         <chr> <chr> <chr>     <dbl> <dbl>
##  1 10     NUMBER        0 <NA>          <NA>  <NA>  10            0     0
##  2 1913   DATE          0 <NA>          <NA>  1913  <NA>          0     0
##  3 1913   DATE          0 <NA>          <NA>  1913  <NA>          0     0
##  4 1913   NUMBER        0 <NA>          <NA>  <NA>  1913          0     0
##  5 1913   NUMBER        0 <NA>          <NA>  <NA>  1913          0     0
##  6 2018   DATE          0 <NA>          <NA>  2018  <NA>          0     0
##  7 2018   DATE          0 <NA>          <NA>  2018  <NA>          0     0
##  8 2018   NUMBER        0 <NA>          <NA>  <NA>  2018          0     0
##  9 2018   NUMBER        0 <NA>          <NA>  <NA>  2018          0     0
## 10 33,000 NUMBER        0 <NA>          <NA>  <NA>  33000         0     0
## # ... with 22 more rows, and 2 more variables: beginOffset <int>,
## #   mention_type <chr>
```

Sentiment of the entire text:

```
nlp_result$documentSentiment
```

```
## # A tibble: 1 x 2
##   magnitude score
##       <dbl> <dbl>
## 1       0.6   0.1
```

Content categories:

List of categories: https://cloud.google.com/natural-language/docs/categories

```
nlp_result$classifyText[[1]]
```

```
## # A tibble: 1 x 2
##   name                                          confidence
##   <chr>                                              <dbl>
## 1 /Jobs & Education/Education/Colleges & Universities       0.99
```

Translation:

Supported languages: https://cloud.google.com/translate/docs/languages

```
translate_result = gl_translate(text, target = "hi") # Hindi
translate_result$translatedText
```

जॉर्जिया स्टेट यूनिवर्सिटी (जॉर्जिया स्टेट, स्टेट, या जीएसयू) अटलांटा, जॉर्जिया में एक सार्वजनिक शोध विश्वविद्यालय है। 1913 में स्थापित, यह जॉर्जिया के चार शोध विश्वविद्यालयों की विश्वविद्यालय प्रणाली में से एक है। यह जॉर्जिया में स्थित उच्च शिक्षा का सबसे बड़ा संस्थान भी है और 52,000 के आसपास विविध छात्र आबादी के साथ देश में शीर्ष 10 में है, जिसमें 2018 तक मुख्य परिसर में लगभग 33,000 स्नातक और स्नातक छात्र शामिल हैं।

```
translate_result = gl_translate(text, target = "zh-TW") # Traditional Chinese
translate_result$translatedText
```

佐治亞州立大學（Georgia State、State 或 GSU）是位於佐治亞州亞特蘭大的一所公立研究型大學。它成立於1913年，是佐治亞大學系統的四所研究型大學之一。它也是佐治亞州最大的高等教育機構，在全國排名前十，截至 2018 年，其多元化的學生人數約為 52,000 人，其中包括市中心主校區的約 33,000 名研究生和本科生。

Text-to-speech:

```r
gl_talk("I am living in Atlanta", gender = "FEMALE", languageCode = "en-US")
```

```
## [1] "output.wav"
```

Go to your working directory, and you should be able to see `output.wav`

Speech-to-text:

Requirements for the audio file:

- Must be a WAV, FLAC, or OPUS file (no MP3)
- Must be 60 seconds or less

```r
speech_result <- gl_speech("output.wav", sampleRateHertz = 24000L)
speech_result
```

```
## $transcript
##               transcript confidence languageCode channelTag
## 1 I am living in Atlanta  0.8349115         <NA>       <NA>
##
## $timings
## $timings[[1]]
##   startTime endTime    word
## 1        0s  0.100s       I
## 2    0.100s  0.300s      am
## 3    0.300s  0.500s  living
## 4    0.500s  0.700s      in
## 5    0.700s  0.800s Atlanta
```

Your turn

Play with the NLP, translation, and text-to-speech APIs using your own text.

Do you find the results reliable?

What are the weakness?

RoogleVision

```r
Sys.setenv("GAR_CLIENT_JSON" = "gcp-service-account-key.json")

library(googleAuthR)
library(RoogleVision)
library(tidyverse)

options("googleAuthR.scopes.selected" =
        "https://www.googleapis.com/auth/cloud-platform")
gar_set_client("gcp-auth.json")

# Alternatively, go through the OAuth2 flow, and save the token
#googleAuthR::gar_auth()
```

Object detection:

Input image: https://dl.dropboxusercontent.com/s/5s8rh8anixemjvn/kitten.png

```r
url = "https://dl.dropboxusercontent.com/s/5s8rh8anixemjvn/kitten.png"
kitten <- getGoogleVisionResponse(
  imagePath=url, numResults=5,
  feature="LABEL_DETECTION")
kitten
```

```
##         mid                 description     score topicality
## 1  /m/04rky                      Mammal 0.9727271  0.9727271
## 2  /m/01yrx                         Cat 0.9703034  0.9703034
## 3  /m/0307l                     Felidae 0.9594564  0.9594564
## 4 /m/07k6w8 Small to medium-sized cats 0.8624860  0.8624860
## 5 /m/023kp2                         Paw 0.8451160  0.8451160
```

Face detection:

Input image: https://dl.dropboxusercontent.com/s/jdu5jvjruqdmd6s/meghan-and-harry.jpg

```
url = "https://dl.dropboxusercontent.com/s/jdu5jvjruqdmd6s/meghan-and-harry.jp
meghan_and_harry = getGoogleVisionResponse(
  imagePath = url,
  feature = "FACE_DETECTION")
glimpse(meghan_and_harry)
```

```
## Rows: 2
## Columns: 15
## $ boundingPoly            <df[,1]> <data.frame[2 x 1]>
## $ fdBoundingPoly          <df[,1]> <data.frame[2 x 1]>
## $ landmarks               <list> [<data.frame[34 x 2]>], [<data.frame[34 x 2]>]
## $ rollAngle               <dbl> 0.9514707, -10.2978520
## $ panAngle                <dbl> -13.96779, 11.38637
## $ tiltAngle               <dbl> -12.27602, -17.64107
## $ detectionConfidence     <dbl> 0.9961154, 0.9999379
## $ landmarkingConfidence   <dbl> 0.6930657, 0.6612572
## $ joyLikelihood           <chr> "VERY_LIKELY", "VERY_LIKELY"
## $ sorrowLikelihood        <chr> "VERY_UNLIKELY", "VERY_UNLIKELY"
## $ angerLikelihood         <chr> "VERY_UNLIKELY", "VERY_UNLIKELY"
## $ surpriseLikelihood      <chr> "VERY_UNLIKELY", "VERY_UNLIKELY"
## $ underExposedLikelihood <chr> "VERY_UNLIKELY", "VERY_UNLIKELY"
## $ blurredLikelihood       <chr> "VERY_UNLIKELY", "VERY_UNLIKELY"
```

Landmark detection:

Input image: https://dl.dropboxusercontent.com/s/1l5ok0eq1f14rs8/taj-mahal.jpg

```
url = "https://dl.dropboxusercontent.com/s/1l5ok0eq1f14rs8/taj-mahal.jpg"
taj_mahal <- getGoogleVisionResponse(
  imagePath = url,
  feature="LANDMARK_DETECTION")
taj_mahal
```

```
##         mid description     score                                  vertices
## 1 /m/0l8cb   Taj Mahal 0.9719254 33, 1258, 1258, 33, 103, 103, 705, 705
##             locations
## 1 27.17470, 78.04207
```

Text detection:

Input image: https://dl.dropboxusercontent.com/s/ug2vzazfectr6h2/atlanta.jpg

```
url = "https://dl.dropboxusercontent.com/s/ug2vzazfectr6h2/atlanta.jpg"
atlanta <- getGoogleVisionResponse(
  imagePath = url,
  feature = "TEXT_DETECTION")
atlanta
```

```
##   locale                  description                         vertices
## 1     en WESTIN\n. THIS IS\nATLANTA\n     237, 760, 760, 237, 7, 7, 328, 328
## 2   <NA>                       WESTIN      239, 359, 356, 237, 7, 17, 44, 34
## 3   <NA>                            . 258, 269, 269, 258, 237, 237, 255, 255
## 4   <NA>                         THIS 351, 541, 542, 352, 191, 188, 250, 253
## 5   <NA>                           IS 567, 639, 640, 568, 192, 191, 249, 250
## 6   <NA>                      ATLANTA 257, 760, 760, 257, 257, 257, 328, 328
```

Your turn

Find a few images and experiment with the Cloud Vision API:

- animals that are less popular
- people with different ages
- people with different skin tones
- Taj Mahal from different angles and distances

Do you find the results reliable? What are the issues?

googleComputeEngineR

```r
Sys.setenv("GCE_AUTH_FILE" = "gcp-service-account-key.json")
options("googleAuthR.scopes.selected" =
        "https://www.googleapis.com/auth/cloud-platform")

library(googleComputeEngineR)

gce_global_project("cis8392")

# zones: https://cloud.google.com/compute/docs/regions-zones/
gce_global_zone("us-east1-b")
```

Create a templated container based VMs:

Templated container based VMs are very flexible and lightweight. They provide an image for an application, which has been pre-configured such as you can just load and use the application right away.

- Available templates from googleComputeEngineR: `rstudio`, `shiny`, `opencpu`, `r-base`, `dynamic`, `rstudio-gpu`, `rstudio-shiny`
- You can choose different machine types (that is, `predefined_type` in our example code on next slide): https://cloud.google.com/compute/docs/machine-types
- It takes time to initialize an VM. In the example below, you will see the IP address of the VM in the console output. You can connect to the VM using your browser, but you may need wait for a couple minutes before it becomes available.

```r
vm <- gce_vm(template = "rstudio", predefined_type = "n1-standard-1",
             name = "rstudio-demo", username = "cis", password = "8392")
```

Cleaning up

To prevent unexpected billing, it is a good idea to remove or shutdown a project from GCP once you finish it.

Make sure that you save all you need from the cloud to your local machine before you shutting down a project!

Go to the **Setting** section of your GCP console:
https://console.cloud.google.com/iam-admin/settings/

You will see the **SHUT DOWN** option there.

Once you shutdown a project, all other resources (Cloud Storage, BigQuery, VM, etc) in the project will be removed--so no more charging.