School of Computing and Information Systems

MAST30034: Applied Data Science

Workshop 2

1 Introduction

The aim of this lab is to continue working on your assignment and setting up your environment to analyse and process the data.

2 Square Binning in R

Square Binning is easy to do in R, you can use the geom_bin2d, in our example, replace this line

```
plotmap <- ggmap(map) + geom_point(aes(x = pickup_longitude, y =
 pickup_latitude), colour="white", size = 0.01, data = mydata, alpha = .5)
```

with

```
plotmap <- ggmap(map) + geom_bin2d(bins=100, data=mydata, aes(x =
 pickup_longitude, y = pickup_latitude))
```

You can vary the number of cells by changing the bins count, you can also set the size of the bins, but using a bins count give you a better idea of the processing load, since you know the number of bins that will be created. Try experimenting with different bin counts. See `https://ggplot2.tidyverse.org/reference/geom_bin2d.html` for further documentation.

3 Hex Binning in R

Hex Binning is also easy to do in R, you can use the geom_hex. First install hexbin:

```
install.packages("hexbin")
```

Then in our example, replace this line

```
plotmap <- ggmap(map) + geom_point(aes(x = pickup_longitude, y =
 pickup_latitude), colour="white", size = 0.01, data = mydata, alpha = .5)
```

with

```
plotmap <- ggmap(map) + coord_cartesian() + geom_hex(bins=100, data=mydata,
 aes(x = pickup_longitude, y = pickup_latitude))
```

Note the addition of `coord_cartesian()` which is a necessary requirement to work with geom_hex.

4 SparkR

These instructions are based on Ubuntu, but a similar method should work on Mac. Installing Spark with R can be a error prone process, we have updated these instructions to make use of SparklyR, which greatly simplifies the process. The following instructions have been tested on the Ubuntu cloud instance, but in principle should work for other platforms.
Assuming you already have R installed, do the following:

```
sudo apt-get install default-jdk
sudo apt-get install openjdk-8-jre-headless
sudo R CMD javareconf
sudo apt-get install r-cran-rjava
sudo apt-get install libgdal-dev libproj-dev
sudo apt-get install libcurl4-openssl-dev
sudo apt-get install libssl-dev
sudo apt-get install libxml2-dev
```

Load R

```
R
```

Install the following R packages:

```
install.packages("sparklyr")
#Install Spark
library(sparklyr)
spark_install(version = "2.1.0")
```

To use Spark

```
Sys.setenv(JAVA_HOME = "/usr/lib/jvm/java-8-openjdk-amd64/jre")
library(sparklyr)
library(dplyr)
sc <- spark_connect(master = "local")
```

4.1 Load Data into SparklyR

Use the following line to load data into SparkR:

```
nyc_taxi <- spark_read_csv(sc, name = "taxi_data", path =
↪  "100k_yellow_2015_05.csv", header = TRUE, delimiter = ",")
```

Note: Be careful with the path, it appears to take the path at the time SparkR was initialised, not a more recently changed working directory path.

4.2 Manual Square Binning by Rounding

The following code will perform square binning in SparklyR and store the result back to the data frame

```
nyc_taxi <- nyc_taxi %>%
  mutate(pickup_latitude = round(pickup_latitude,3))%>%
  mutate(pickup_longitude = round(pickup_longitude,3))%>%
  sdf_register("nyc_taxi")
```

If you want so save that data you can use the following, which will save it to the directory in a series of partitioned files, however, be careful, as this will require a lot of spacewhen using a larger input file.

```
spark_write_csv(nyc_taxi,"rounded",header=TRUE,delimiter=",", mode="overwrite")
```

If you want to read the data back into spark, just use the folder as path in the texttt spark_read_csv.
To summarise the data call the following:

```
nyc_taxi_summary <- nyc_taxi %>%
group_by(pickup_latitude, pickup_longitude) %>%
  summarise(n=n()) %>%
  sdf_register("nyc_taxi_summary")
```

If you want to check the output use `head(nyc_taxi_summary)` which will show just the first few rows. You should see something like the following (the decimal places are truncated in this view):

```
pickup_latitude pickup_longitude     n
            <dbl>            <dbl> <dbl>
1            40.7            -74.0     2
2            40.8            -74.0     1
3            40.8            -74.0     1
4            40.7            -74.0     3
5            40.8            -74.0     3
6            40.7            -74.0     3
```

You can save the summary using

```
spark_write_csv(nyc_taxi_summary,"summary",header=TRUE,delimiter=",", mode="overwrite")
```

The squares can be plotted with the following

```
plotmap <- ggmap(map) + geom_point(aes(x = pickup_longitude, y =
                                       pickup_latitude, colour=n,
                                       fill=n), data = nyc_taxi_summary,
                                       shape=22, size=0.25)
ggsave("plot.png")
```

Note: we now set the the colour and fill values to the count column in order to scale the colours. You can adjust size to expand the points up to the point of overlap, or calculate the exact sizing based on the rounding performed on the GPS co-ordinates. A more accurate display can be achieved by plotting polygons using the calculated GPS co-ordinates. You might also want to adjust the colour scales to make it more distinct, or use the summary data in a heatmap directly, now that it is much smaller in size.