Project Proposal

Analysis of *On-Street Car Parking* in the city of Melbourne

Name - Simran Singh Gulati
Student ID - 31125301
Tutor - Jeffery Chieh Liu

Questions

1. How does the availability compare with the demand?
2. How do the parking restrictions compare across the city and on different days of the week?
3. What role do restrictions play in ensuring a *fair* chance of finding a spot?

Data Sources

A. On Street Parking Bay Sensors
B. On Street Car Park Restrictions
C. On Street Parking Bays

The data source A will allow me to answer the first question. While B would aid in answering the second question. And the third can be answered with combination of both the datasets (A and B).

Lastly the data source C can help visualise all the explorations spatially, so it plays a preliminary role in visualising all the scenarios (1, 2 and 3)

Description of Data Sources

1. Tabular Data : 3459 rows X 6 columns
   Each row contains information about the sensor - id, location and status (occupied or not)
   https://data.melbourne.vic.gov.au/Transport/On-street-Parking-Bay-Sensors/vh2v-4nfs

2. Tabular Data : 4518 rows X 62 columns
   Each row contains restrictions (information) specific to a parking bay
   https://data.melbourne.vic.gov.au/Transport/On-street-Car-Park-Bay-Restrictions/ntht-5rk7/data

3. Spatial Data
   It contains spatial polygons to mark parking bays on a map
   https://data.melbourne.vic.gov.au/Transport/On-street-Parking-Bays/crvt-b4kt

Street Parking across the city of Melbourne

Section 1 — Exploration

Name -	Simran Singh Gulati
Student ID -	31125301
Lab Number -	Online 03 - Wednesday, 6-8 pm
Tutor Name -	Jie (Lewis) Liu, Jeffery Chieh Liu

Introduction

The project aims at surveying the availability of on street parking locations across the city of Melbourne, and investigating the restrictions in each parking spot.

At first, we will try to compute *if and how* the availability competes with demand for the parking spots. This would allow us to rule out if we have enough parking locations to accomodate the ongoing traffic across our city. Secondly, it would set up a basis for our next probe.

Once we have an idea of the demand for the parking spots, we proceed towards scrutinisation of the Parking Restrictions in those spots. The very idea of imposing restrictions is to ensure that there's fair use of public space and that every car gets a fair chance of finding a spot but we want to check how do the parking rules change and what factors may influence these rules across varied locations and on different days of the week. Additionally, we may also take into account the weekend traffic and the demand on public holidays.

For instance, the demand for a parking spot would be significantly greater in populous and busy areas like Flinders or CBD. Basically, the business centres or universities would draw greater car traffic and thus the resulting requirement for parking locations in that area. Now another aspect is, that these locations are busier on the weekdays and there might be some other areas like the beaches or leisure spots including restaurants and clubs which tend to draw attention only on the weekends. This factor contributes greatly in designing the parking restrictions.

All in all, how do these tiny details affect and influence the authorities to compose regulations, is what we are interested in.

Data Wrangling and Checking

The data comprises of three separate files :

A. On Street Parking Bay Sensors — Tabular Data

Each parking bay has an electronic device installed beneath, which senses if a car is parked in that spot or not. This files contains the information collected by these sensors in long form. The columns which are relevant to us include the bay id, location co-ordinates of that bay and its status (occupied or not).

Source : https://data.melbourne.vic.gov.au/Transport/On-street-Parking-Bay-Sensors/vh2v-4nfs

B. On Street Car Park Restrictions — Tabular Data

Additionally, the parking locations hold some restrictions that apply to them. This file contains those rules in the wide form. There are 62 columns which state how many (up-to 6) restrictions apply to each particular spot. Each restriction is associated with a description attribute, start time, end time, permitted duration, extended duration for disabled drivers, exemption and whether the restriction applies of Public Holidays.
Each row can be linked to the first dataset using the bay id.

Source : https://data.melbourne.vic.gov.au/Transport/On-street-Car-Park-Bay-Restrictions/ntht-5rk7/data

C. On Street Parking Bays — Spatial Data

It contains spatial polygons to visualise parking bays on a map. And can be linked to first dataset using the unique marker id.

Additionally, it can also be linked to parking meters to compute the parking fair/charges but that is outside the scope of this report as we are only concerned about the exploration phase. Thus the datasets B and C can be joined with A but not directly with each other.

Source : https://data.melbourne.vic.gov.au/Transport/On-street-Parking-Bays/crvt-b4kt

Now that we know what kind of data we posses, we begin exploring each dataset and reformat into the desired form. Beginning with the first *csv* file, we first import it into R Studio and examine the columns and their data types.

Image 1 — dataset 1

```
> sensors_data <- read.csv("On-street_Parking_Bay_Sensors.csv")
> glimpse(sensors_data)
Rows: 3,567
Columns: 6
$ bay_id       <int> 1270, 6840, 5074, 1690, 4315, 7873, 8913, 1315, 7052, 6660, 1121, 1287, 5135, 3201, 1280, 9006, 2409, 2570, 1288, 6307, 1285, 1726, 3371, 8949,…
$ st_marker_id <chr> "4462E", "14040E", "9854N", "2240N", "C6966", "4472EA", "18000E", "1315W", "4664Ea", "13415W", "C1158", "4479W", "9948N", "5457W", "4472E", "C7…
$ status       <chr> "Present", "Unoccupied", "Unoccupied", "Unoccupied", "Unoccupied", "Unoccupied", "Unoccupied", "Unoccupied", "Unoccupied", "Present", "Present"…
$ location     <chr> "(-37.813703354077106, 144.95461916833347)", "(-37.796658516509055, 144.95814457090097)", "(-37.82520231330287, 144.9698326288035)", "(-37.8162…
$ lat          <dbl> -37.81370, -37.79666, -37.82520, -37.81628, -37.80939, -37.81308, -37.80341, -37.81708, -37.80570, -37.81966, -37.81375, -37.81296, -37.82591, …
$ lon          <dbl> 144.9546, 144.9581, 144.9698, 144.9599, 144.9562, 144.9543, 144.9601, 144.9587, 144.9595, 144.9498, 144.9601, 144.9540, 144.9674, 144.9544, 144…
> |
```

As evident from the screenshot, there are are 3,567 rows. Each row represents a different sensor and each sensor can be uniquely identified by the bay_id or the st_marker_id. Though both these attributes can distinguish any two sensors from each other but we are interested in using only the bay_id.

Firstly, keeping 2 unique attributes for each row does not make sense. Secondly, the bay_id is also an attribute in the second dataset which makes it more suitable to be used than the st_marker_id. Thirdly, the default datatype for bay_id is INT (integer) unlike the st_marker_id. Now, this fact can be quite subjective but what I understand by looking at the data — the *bay_id* is following a number sequence and sensors in close proximity can be easily spotted or looked up when using a number system approach (this is referred to in detail in the exploration section of the report when we visualise the sensors geographically). Additionally, looking at a hypothetical scenario, when the authorities are to install new sensors it's easier to keep track using the bay_id instead of the st_marker_id.

Similarly the Status is in Char form and we typecast the column values to Factors with values 1 for parking being available and 0 for not available. We use the *is.na* and *unique* functions to check for empty values or any duplications but fortunately the entries are distinct, clean and well formatted. One of the reason for this could be that data is collected from an electronic device (parking sensor) whose whole purpose is to fetch this information and report it to the the supervising authority. But unlike this file, our next dataset has been put up and constructed by humans and is expected to have some discrepancies, so let's see how it goes.

Image 2 — dataset 2

```
> restrictions_data <- read.csv("On-street_Car_Park_Bay_Restrictions.csv")
> glimpse(restrictions_data)
Rows: 4,518
Columns: 62
$ BayID           <int> 3101, 4919, 6664, 4972, 8634, 3108, 8167, 8587, 3103, 3928, 8498, 3105, 7978, 8099, 4893, 3947, 8083, 6849, 3106, 7981, 7966, 8152, 8089,…
$ DeviceID        <int> 27429, 21769, 21773, 21813, 28003, 26925, 25546, 27956, 27141, 27594, 27876, 24680, 25357, 25478, 21756, 24656, 25462, 24688, 24872, 2536…
$ Description1    <chr> "1/2P M-SUN 7:30AM-11PM", "2P MTR M-F 7:30-18:30", "2P MTR M-F 7:30-18:30", "1P MTR M-F 7:30-18:30", "2P RPA 1B 7:30-23:00 M-SUN", "1/2P …
$ Description2    <chr> "", "", "", "", "", "", "", "", "", "4P MTR SAT 7:30-12:30", "", "", "", "", "", "4P MTR SAT 7:30-12:30", "", "4P TKT A SAT 7:30-12:30", …
$ Description3    <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "1P SUN 7:30-18:30", "", "2P SUN 7:30…
$ Description4    <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Description5    <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Description6    <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ DisabilityExt1  <int> 60, 240, 240, 120, 240, 60, 120, 240, 60, 480, 240, 60, 240, 240, 240, 480, 120, 480, 60, 240, 240, 120, 240, 240, 240, 0, 120, 60, 240, …
$ DisabilityExt2  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, 480, NA, NA, NA, NA, NA, 480, NA, 480, NA, NA, NA, NA, NA, NA, NA, 120, NA, 240, NA, NA, NA, NA, 240,…
$ DisabilityExt3  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 120, NA, 240, NA, NA, NA, NA, 120, NA…
$ DisabilityExt4  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ DisabilityExt5  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ DisabilityExt6  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ Duration1       <int> 30, 120, 120, 60, 120, 30, 60, 120, 30, 240, 120, 30, 120, 120, 120, 240, 60, 240, 30, 120, 120, 60, 120, 120, 120, 15, 60, 30, 120, 120,…
$ Duration2       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, 240, NA, NA, NA, NA, NA, 240, NA, 240, NA, NA, NA, NA, NA, NA, NA, 60, NA, 120, NA, NA, NA, NA, 120, …
$ Duration3       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 60, NA, 120, NA, NA, NA, NA, 60, NA, …
$ Duration4       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ Duration5       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ Duration6       <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ EffectiveOnPH1  <int> 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,…
$ EffectiveOnPH2  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, 0, NA, NA, NA, NA, NA, 0, NA, 0, NA, NA, NA, NA, NA, NA, NA, 0, NA, 0, NA, NA, NA, NA, 0, NA, NA, NA,…
$ EffectiveOnPH3  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 0, NA, 0, NA, NA, NA, NA, 0, NA, NA, …
$ EffectiveOnPH4  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ EffectiveOnPH5  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ EffectiveOnPH6  <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ EndTime1        <chr> "23:00:00", "18:30:00", "18:30:00", "18:30:00", "23:00:00", "23:00:00", "23:00:00", "23:00:00", "23:00:00", "18:30:00", "23:00:00", "23:0…
$ EndTime2        <chr> "", "", "", "", "", "", "", "", "", "12:30:00", "", "", "", "", "", "12:30:00", "", "12:30:00", "", "", "", "", "", "", "", "19:30:00", "…
$ EndTime3        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "18:30:00", "", "18:30:00", "", "", "…
$ EndTime4        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ EndTime5        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ EndTime6        <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Exemption1      <chr> "", "", "", "", "Resident Permit Area 1B", "", "", "Resident Permit Area 1B", "", "", "Resident Permit Area 1B", "", "", "", "", "", "", …
$ Exemption2      <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Exemption3      <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Exemption4      <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Exemption5      <chr> "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "", "…
$ Exemption6      <lgl> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ FromDay1        <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
$ FromDay2        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, 6, NA, NA, NA, NA, NA, 6, NA, 6, NA, NA, NA, NA, NA, NA, NA, 6, NA, 1, NA, NA, NA, NA, 1, NA, NA, NA,…
$ FromDay3        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, 0, NA, 0, NA, NA, NA, NA, 0, NA, NA, …
$ FromDay4        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ FromDay5        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ FromDay6        <int> NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, NA, N…
$ StartTime1      <chr> "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:30:00", "07:3…
```

Having a bird's eye view at the data made me think that it has a lot of empty values. While yes, the blank values need to be replaced by something but there's a reason that those cells are empty. What I intend to say is, having a 0 or NA in place of empty cells would have been better but the values being empty does not mean the data is incomplete or irrelevant for our investigation. And the reasoning to my claim can be explained by looking at the column descriptions and what they intend to convey.

The first column for each row states the bay_id, which obviously is well populated and doesn't draw our concern. Same is the case with the device ID. Rest of the columns are very specific and need to examined carefully. The attributes Description, Duration, Exemption, FromDay, ToDay etc. range from 1 to 6 which implies that there can be up-to 6 restrictions for each parking spot. Reasons include, separate rules for weekdays versus weekends and busy hours versus the non peak time or even public holidays.

Thinking on these lines, such multi tier restriction would only apply to the parking bays with highly varied traffic trends. While bays which experience monotonic trends or which are usually empty wouldn't need such complex regulations. And this is the reason Description1 has no empty values but Description2 has a lot of empty values. Extending these results, the columns associated with Description2 would also be empty — such as Duration2, FromDay2, ToDay2, EndTime2, and so on.

To better convey my point, I'll attach a code snapshot which is a logical proof for the statement. The code reveals total number of rows with NA values in a specific column. We see that FromDay1 has 0 null values, while FromDay2 has 1453 empty values which means 1453 parking bays don't have a second restriction at all. And 3873 parking bays don't have a third restriction and so on.

You might be wondering why the FromDay2 column was chosen and not the Description2. Interestingly, the Description columns have CHAR format which puts blank spaces instead of NA values, while other columns with INT type, like this one, are pretty helpful for such analysis.

```
> sum(is.na(restrictions_data$FromDay1))
[1] 0
> sum(is.na(restrictions_data$FromDay2))
[1] 1453
> sum(is.na(restrictions_data$FromDay3))
[1] 3873
> sum(is.na(restrictions_data$FromDay4))
[1] 4314
> sum(is.na(restrictions_data$FromDay5))
[1] 4452
> sum(is.na(restrictions_data$FromDay6))
[1] 4499
>
```

Image 3— na values

To keep our experiment concise, I decided to keep only the primary restrictions. Yes, leaving out rest of the data would not be entirely fair but given the fact that most of the bays do not have a secondary or tertiary restrictions it seems a reasonable trade off. This streamlines the data and gives us a broader viewpoint for the exploration phase.

All in all, we keep the columns associated with Description1, perform appropriate type conversions, replace blank values with 0 and drop irrelevant entries.

```
> restrictions_data_mod <- restrictions_data %>% select(BayID, Description1, DisabilityExt1, Duration1, EffectiveOnPH1, EndTime1, Exemption1, FromDay1, ToDay1, Typ
> glimpse(restrictions_data_mod)
Rows: 4,518
Columns: 10
$ BayID          <int> 3101, 4919, 6664, 4972, 8634, 3108, 8167, 8587, 3103, 3928, 8498, 3105, 7978, 8099, 4893, 3947, 8083, 6849, 3106, 7981, 7966, 8152, 8089,…
$ Description1   <chr> "1/2P M-SUN 7:30AM-11PM", "2P MTR M-F 7:30-18:30", "2P MTR M-F 7:30-18:30", "1P MTR M-F 7:30-18:30", "2P RPA 1B 7:30-23:00 M-SUN", "1/2P …
$ DisabilityExt1 <int> 60, 240, 240, 120, 240, 60, 120, 240, 60, 480, 240, 60, 240, 240, 240, 480, 120, 480, 60, 240, 240, 120, 240, 240, 240, 0, 120, 60, 240, …
$ Duration1      <int> 30, 120, 120, 60, 120, 30, 60, 120, 30, 240, 120, 30, 120, 120, 120, 240, 60, 240, 30, 120, 120, 60, 120, 120, 120, 15, 60, 30, 120, 120,…
$ EffectiveOnPH1 <int> 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1,…
$ EndTime1       <chr> "23:00:00", "18:30:00", "18:30:00", "18:30:00", "23:00:00", "23:00:00", "23:00:00", "23:00:00", "18:30:00", "23:00:00", "23:0…
$ Exemption1     <chr> "", "", "", "", "Resident Permit Area 1B", "", "", "Resident Permit Area 1B", "", "", "Resident Permit Area 1B", "", "", "", "", "", "", …
$ FromDay1       <int> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,…
$ ToDay1         <int> 0, 5, 5, 5, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, 5, 5, 0, 5, 0, 0, 0, 0, 0, 5, 5, 5, 0, 6, 0, 0, 0, 0, 6, 0, 0, 0, 0, 0, 0, 0, 5, 0, 5, 6, 0, 0,…
$ TypeDesc1      <chr> "1/2P", "2P Meter", "2P Meter", "1P Meter", "2P", "1/2P", "1P", "2P", "1/2P", "4P Meter", "2P", "1/2P", "2P Ticket A", "2P", "2P Meter", …
> write.csv(restrictions_data_mod, "parking_restrictions.csv", row.names = TRUE)
```

Image 4— final data source

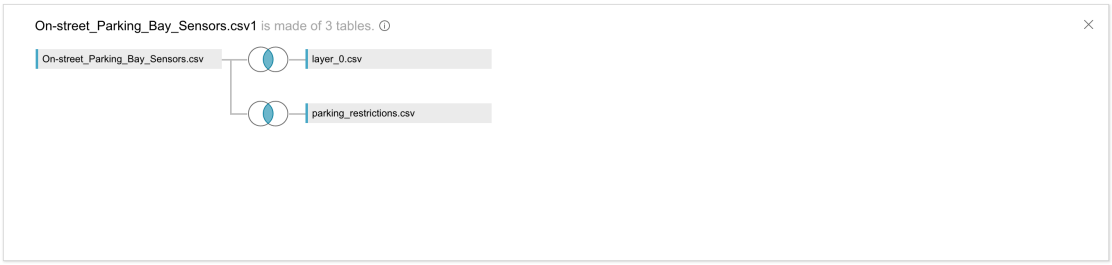
So this is how the data looks like after all the processing and was exported as a new *csv* file which will set up a base for our visualisations.

Note - The column DeviceID uniquely identified each sensor but so did the BayID, thus keeping only one of them made sense as discussed already. More importantly, the attribute BayID allows us to join the two datasets which the reader can see in the later sections of the report.

Lastly, the first dataset contains device information about 3,567 sensors while second dataset has restrictions associated with 4,518 sensors. So to streamline their connection we perform an inner join which returned a total of 3,322 .

Image 5— final data source

Finally our data setup looks like —



Sort fields A to Z ascending | Show aliases | Show hidden fields | 1,000 rows

bay_id	bay_id (layer_0.csv)	BayID	Description1	DisabilityExt1	Duration1	EffectiveOnPH1	EndTime1	Exemption1	F1	FromDay1	last_edit
2298	2298	2298	1P MTR M-F 9:30-19:30	120	60	0	30/12/1899 7:30:00 PM	null	2,329	1	20,200,722
6919	6919	6919	P10 M-SUN 0:00-23:59 B	20	10	1	30/12/1899 11:59:59 PM	null	3,760	1	20,200,722
1129	1129	1129	2P MTR M-SAT 7:30-20:30	240	120	0	30/12/1899 8:30:00 PM	null	4,319	1	20,200,722
5032	5032	5032	2P MTR M-F 7:30-18:30	240	120	0	30/12/1899 6:30:00 PM	null	599	1	20,200,722
6643	6643	6643	LZ 30M M-SUN 7:30-19:30	0	30	0	30/12/1899 7:30:00 PM	null	4,487	1	20,200,722
741	741	741	2P MTR M-SAT 7:30-20:30	240	120	0	30/12/1899 8:30:00 PM	null	1,196	1	20,200,722
6353	6353	6353	2P MTR M-SAT 7:30-18:30	240	120	0	30/12/1899 6:30:00 PM	null	4,241	1	20,200,722
3514	3514	3514	LZ 30MIN 722 >	0	30	1	30/12/1899 11:59:59 PM	null	147	1	20,200,722
3401	3401	3401	2P RPA M-F 7:30-18:30	240	120	0	30/12/1899 6:30:00 PM	Residents Per...	683	1	20,200,722
2626	2626	2626	1P MTR M-F 7:30-16:00	120	60	0	30/12/1899 4:00:00 PM	null	2,941	1	20,200,722
4805	4805	4805	2P MTR M-F 7:30-18:30	240	120	0	30/12/1899 6:30:00 PM	null	2,550	1	20,200,722
4032	4032	4032	4P MTR M-F 7:30-18:30	480	240	0	30/12/1899 6:30:00 PM	null	3,058	1	20,200,722
3011	3011	3011	1/2P M-F 7:30-18:30	60	30	0	30/12/1899 6:30:00 PM	null	3,575	1	20,200,722
1470	1470	1470	2P DIS ONLY 7.30 - 20.30	0	120	1	30/12/1899 8:30:00 PM	null	1,362	1	20,200,722
6378	6378	6378	LZ 30M M-SUN 7:30-18:30	0	30	0	30/12/1899 6:30:00 PM	null	2,757	1	20,200,722
3282	3282	3282	4P MTR M-F 7:30-18:30	480	240	0	30/12/1899 6:30:00 PM	null	1,747	1	20,200,722
2441	2441	2441	2P MTR M-SAT 7:30-20:30	240	120	0	30/12/1899 8:30:00 PM	null	1,167	1	20,200,722

Data Exploration

The first instinct when one sees spatial data is to plot the location coordinates on a map, and so do we —

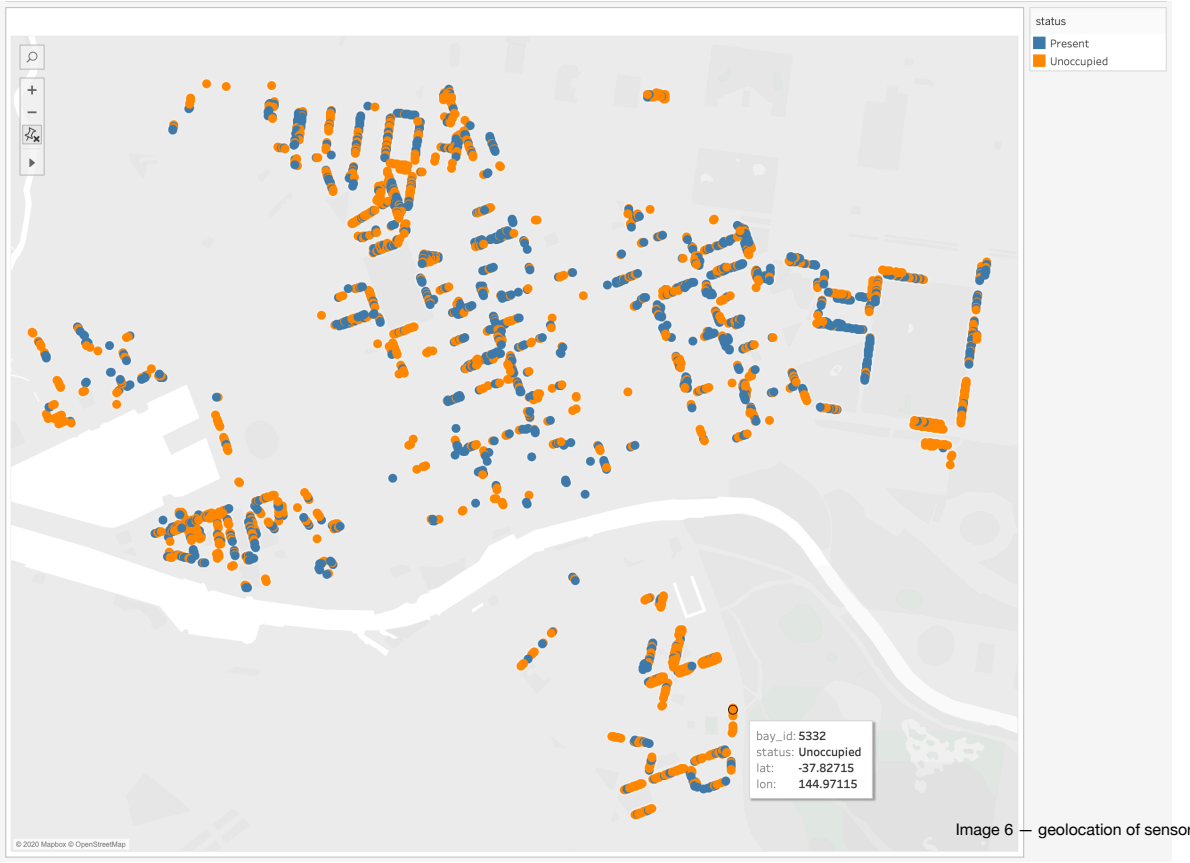


Image 6 — geolocation of sensors

The map above shows the parking bays spread across the city of Melbourne. Each circle represents an individual sensor and the colour is used denote if the spot is occupied or not. Additionally, hovering over a sensors reveals the unique bay id for that particular parking bay. Having this information, we looked up the surrounding sensors and found that parking spots in close proximity are numbered according to a sequence. For instance, we zoomed in on the bay id 5332, and the sensor placed vertically above this one has the bay id 5333; while those placed towards the south had bay id(s) 5331, 5330, 5329 which continued till 5321. Though this information was not known before hand but now that it is decoded, we realise that it is the most logical approach to number the parking bays.

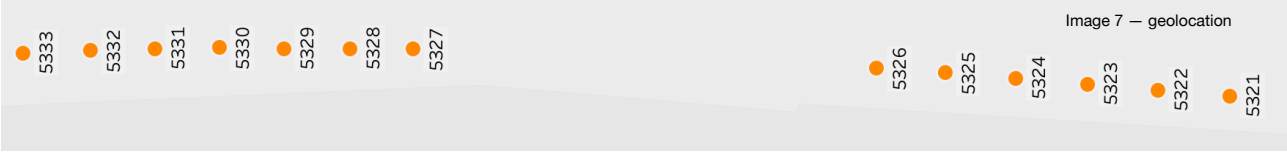


Image 7 — geolocation

(The image is rotated 90 degreed to the left)

Going back to the map as a whole, we see the colour orange (Unoccupied Parking Spots) dominating the overall distribution which makes us think that almost all the bays are vacant and this does not make a lot of sense.

Thinking on these lines we curated a Dashboard style representation which allows the user to dynamically control the sensors being visualised.

As evident when we select the occupied bays we still see a lot cars being parked. Thus our suspicion was correct and a reason for this misinterpretation could be, that there are over 3000 sensors to be visualised and the map is intensively scaled down to show all of them. And the overlapping of circles mis conveys the actual results.

The stacked bar on the right hints that there's near equal distribution of both the types, with slightly more unoccupied spots than the occupied ones.

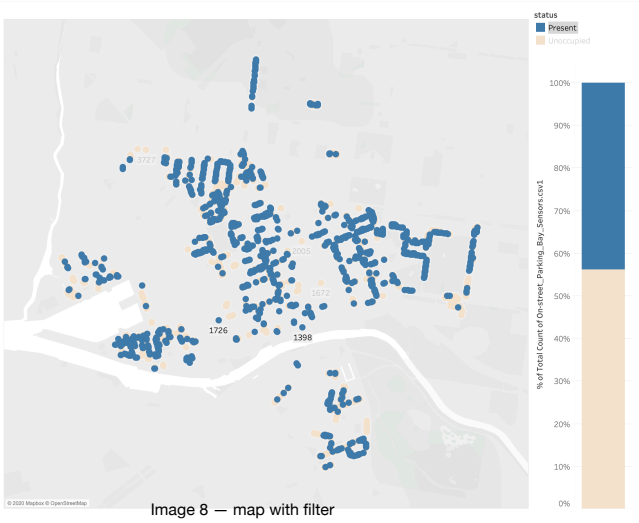


Image 8 — map with filter

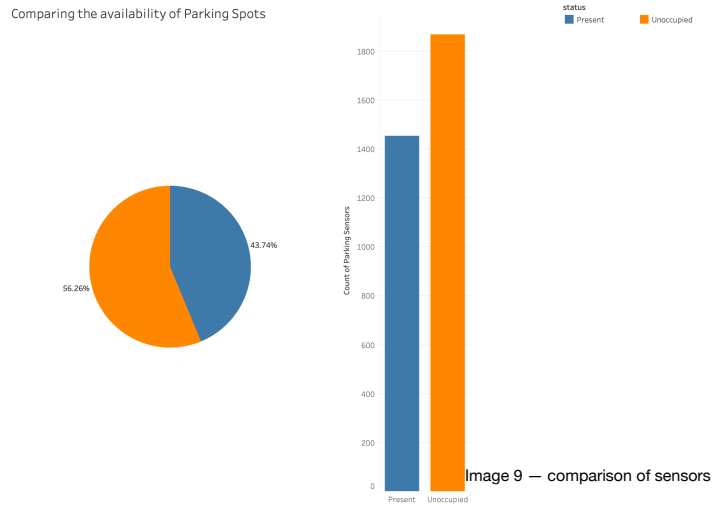


Image 9 — comparison of sensors

To better compare the availability and the demand for parking spots, we produced a pie chart to show ratio distribution of each type and a bar graph to point out the exact numbers.

The figure reveals that there are only 44% bays that have been occupied and the rest are freely available.

Though percentages are helpful for comparisons or generalisation, the bar graph on the other hand gives us the total count of each type, revealing that there are close to 1900 parking bays being empty.

I was expecting around 80% of bays to be occupied but the actual numbers are not even close, probably due to the city wide lockdown and the COVID situation. Continuing with the exploration phase, such reasonings are discussed in the *Conclusion* section of the report.

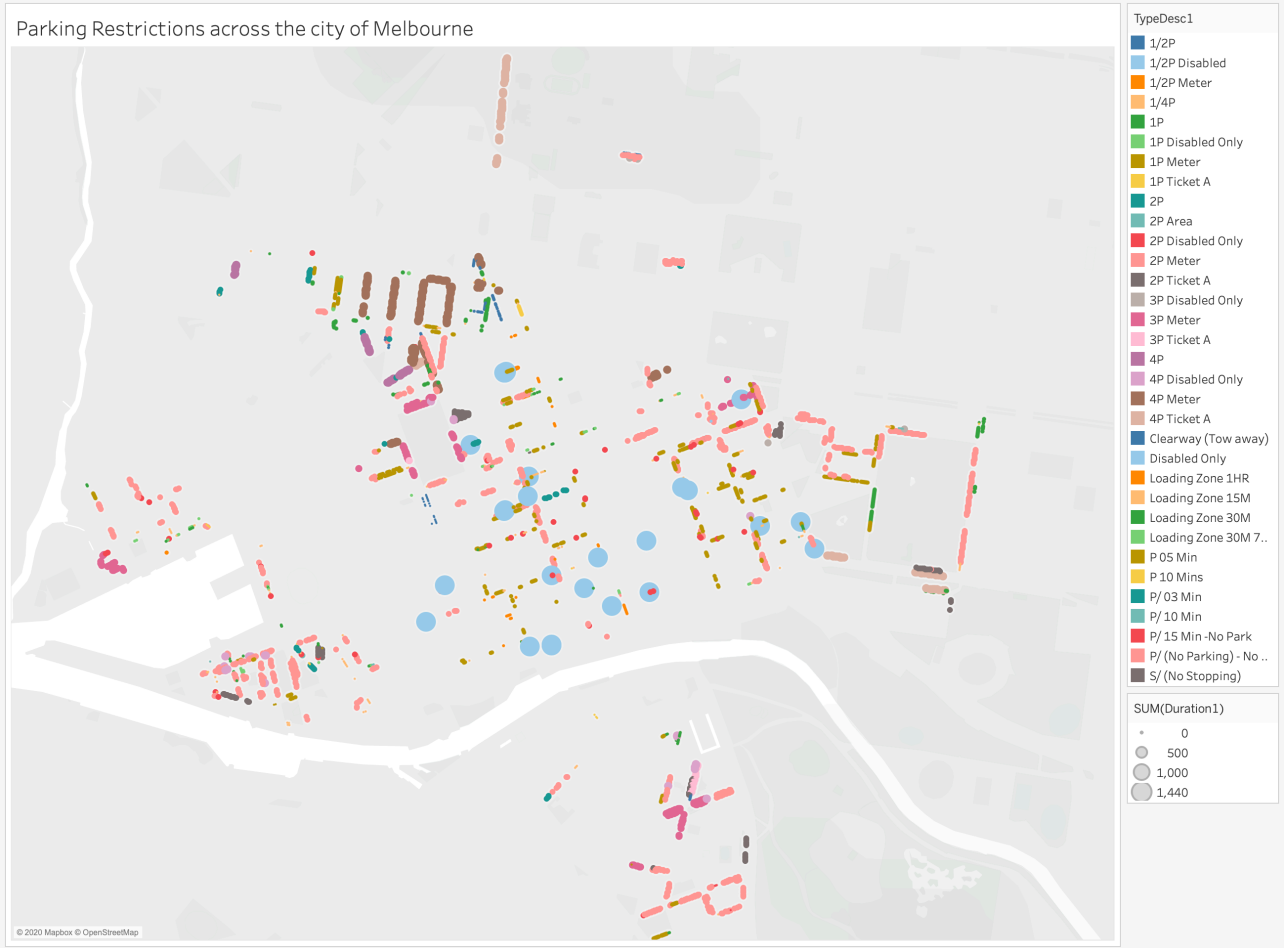
Now we level up the process by introducing the primary restrictions for each sensor.

As discussed earlier the restrictions on a bay vary depending upon a number of factors. Thus we begin by inspecting the different restriction types —

Image 10 — unique restrictions

```
[1] 185
> unique(df$Description1)
  [1] "1P MTR M-SAT 7:30-18:30"                  "2P MTR M-F 7:30-16:00"                  "2P MTR M-SAT 7:30-20:30"
  [4] "1/4P 07.30 - 23.00"                       "LZ 30M M-F 7:30-19:30"                  "P DIS ONLY 20.30 - 23.59"
  [7] "2P TKT A M-SAT 7:30-20:30"                "LZ 15MINS 7.30am - 7.30pm"              "3P MTR M-SAT 7:30-20:30"
 [10] "2P DIS ONLY 7.30 - 20.30"                 "1P MTR M-F 9:30-18:30"                  "1P MTR M-F 7:30-16:30"
 [13] "4P DIS 7.30-8.30"                         "LZ 15M M-SAT 7:30-19:30"                "2P DIS 7.30AM TO 8.30PM"
 [16] "2P MTR M-F 9:30-20:30"                    "2P MTR M-F 7.30am - 4.30pm"             "1/2P DIS 7.30 - 4.30 M-F"
 [19] "LZ 15M M-F 7:30-16.30"                    "LZ 15M M-F 9:30-19:30"                  "2P DIS 7.30 - 8.30PM"
 [22] "P DIS AOT 20:30 TO 23:59"                 "LZ 15M M-F 7:30-19:30"                  "1/2P MTR M-SAT 7.30 TO 18.30"
 [25] "P/ (NO PARKING 621) M-SUN 0.00 TO 11.59"  "LZ 30MINS -722 - M-F 7.30-19.30"        "1/2P MTR M-F 7:30-16:30"
 [28] "P DISABLE AOT 0:00 TO 7:30"               "P 10 MINUTES"                           "CW TOW M-F 7:00-10:00"
 [31] "1P MTR M-F 9:30-16:00"                    "LZ  15M M-F 9:30-16:00"                 "2P MTR M-F 9:30-16:00"
 [34] "1/4P M-F 9:30-16:00"                      "1/4P no time or days"                   "L/Zone 30MINS 7.30 - 6.30PM"
 [37] "P/ 10 M-SUN 0:00-11:59"                   "1/2 P MTR < > 6.00AM TO 11.00PM"        "1P Meter 6.00-16.30"
 [40] "1/4P M-F 7:30-19:30"                      "LZ 30mins 7.30 23.00"                   "P DIS AOT 00.00-07.30"
 [43] "2P DIS M-SUN 7:30-19:30"                  "1/4P M-SUN 7:30-18:30"                  "P10 M-SUN 0:00-23:59"
 [46] "2P DIS ONLY M-SAT 7:30-20:30"             "P DIS AOT"                              "1P MTR M-SAT 7:30-19:30"
 [49] "3 P Dis 730am - 8.30pm"                   "P/ 3 Min (No Parking 621) Mo-Su"        "LZ 30min 7.30am-8.30pm M-Fr"
 [52] "3P MTR M-F 9.30 TO 16.30"                 "3P MTR M-SAT 7.30 - 18.30"              "2P MTR M-F 7.30 TO 16.30"
 [55] "2P DIS M-SUN 0:00-23:59"                  "2P MTR SAT 7:30-20:30"                  "2P MTR M-SAT 19:30-20:30"
```

Though the dataset description mentions the column to be a *human readable* description of the restrictions in that bay, I still feel the values stored within are not easily interpretable. Stressing on each entry (for ex - "1P MTR M-SAT 7:30 - 18:30") I feel the later part is self explanatory which gives us the days (Monday to Saturday) and time (7:30 to 18:30) that each particular restriction is applicable on but the first two words such as "1P MTR" and "2P DIS" are yet to be figured out. Looking up online we found that "1P MTR" refers to 1 Hour Parking Meter and (example 2) "2P DIS ONLY" refers 2 Hours Parking Meter for the Disabled drivers Only.



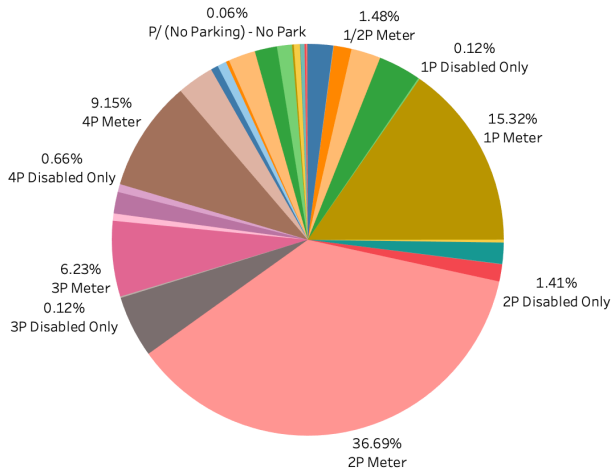


Image 12— pie chart - restriction types

I believe the days and duration of each restriction can be picked up from columns such as FromDay1, ToDay1, Duration1,

Image 11— restrictions types

"1/4P"	"Loading Zone 30M"
"Loading Zone 15M"	"3P Meter"
"1/2P Disabled"	"1/2P Meter"
"P 10 Mins"	"Clearway (Tow away)"
"P/ 03 Min"	"S/ (No Stopping)"
"1P Disabled Only"	"4P"
"4P Ticket A"	"3P Ticket A"
"2P Area"	"Loading Zone 1HR"

StartTime1 and EndTime1. Also, the column TypeDesc1 has way better explanation for the restrictions —

Plus, these columns have already been converted to appropriate data types in the preprocessing phase. One may argue that this step could have been put in the Data Checking phase but that's one aspect of the field of Data Science where the wrangling and exploration steps sometimes overlap and the Analyst has to loop back and fourth for best results. Similarly we found this anomaly in the Exploration step, went back and dropped the column from dataset and now we returned for further exploration. The reason parsing the second dataset is not as smooth as the first one is, that this file has been composed by humans unlike the sensor generated first file.

We see there are 33 different types of restrictions and I wish to know how are the bays distributed among these categories. Hence a pie chart (next page) of the spread is generated to see the number of bays falling in each variation. The chart reveals that 37% of bays have a 2 hour restriction, followed by 15% bays having 1 hour restriction and 9% having a 4 hour window for parking your vehicle.

We observe 2 categories acquiring 50% of the pie chart which implies that rest of 31 types are cramped into other half of the circle. Of these less frequent 31 categories, 28 have lower than 1% (each) of contribution. Well, most of these restrictions differ due to the number of hours they permit for a vehicle to be parked and rest include special conditions like "No Parking" at all, or those for Disabled people. This means we can extend our analysis by directly incorporating the Duration1 attribute.

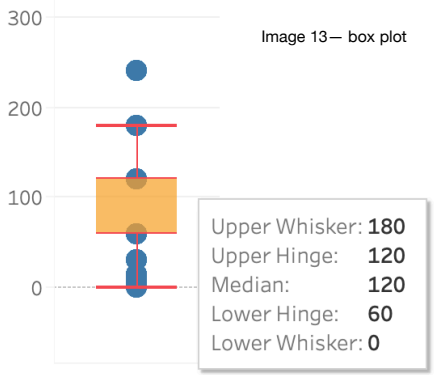


Image 13— box plot

Image 14— box plot stats

```
> summary(df$Duration1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
    0.0    60.0   120.0   126.9   120.0  1440.0
```

The Duration1 column stores the time permitted for parking in each bay. Since there are more than 3000 rows, and most fall into 3 categories, we decided to analyse the distribution using a box plot. The Y Axis labels the total count and whiskers mark the 3 Quartiles.

It confirms that 50% of bays have less than 100 minutes of permitted time. A fourth of bays have 60 minutes or less permitted time. Similarly one fourth of bays have more than 120 minutes allowed for parking. There are a few outliers, allowing 180, 240 or even 1440 minutes (visible in R Summary but not in

boxplot). And many outliers allowing less than 60 minutes of parking time; those include highly restricted areas with 3, 10 or 30 minutes of permitted time. We can also see areas with 0 permitted minutes which points towards the No Parking Zones and Ticket Areas.

We try to put together all the facts discussed above —
As evident, the circles represent parking bays on a map. The colour is used to denote the meter type and the radius of circle is proportionate to the time permitted in that bay. We see the blue circles shine out due to their exceptionally big size. It refers to parking bays for Disabled drivers allowing parking time of up to 24 hours (1440 minutes — outlier). Followed by brown (4P meter — 4 hours) and peach (2P meter — 2 hours).

We see same colours aggregating in certain areas, like there are lot of brown sensors in the North. Blue sensors are concentrated towards the centre of the city, and are considerably less in number. While the sensors in peach colour are present in large numbers and evenly spread across the map.

When we try to relate the above mentioned observations from the map, we realise that blue refers to parking bays for the disabled and are quite less in number. Plus the availability of such spots is evenly spread in the city centre and no two blue sensors are close to each other. Thereby maintaining efficient use of public space as there are less number of disabled drivers.

Next the peach colour represents 37% (from pie-cart) of sensors and can be seen on the map in great numbers. These are termed the standard “2 hour” parking spots available for the general public. Thereby 2P meter can be said to be the common choice by the supervising authority.

Towards the West, we see an accumulation pink sensors which mark the 3P meters with 3 hours of permitted time. Lastly the green sensors refer to 1P meters and can be found in good numbers in both North and East part of the map. The image may look cramped due to a large number of sensors in a very concise map but selecting each sensor type from the right panel is quite helpful to make these claims —

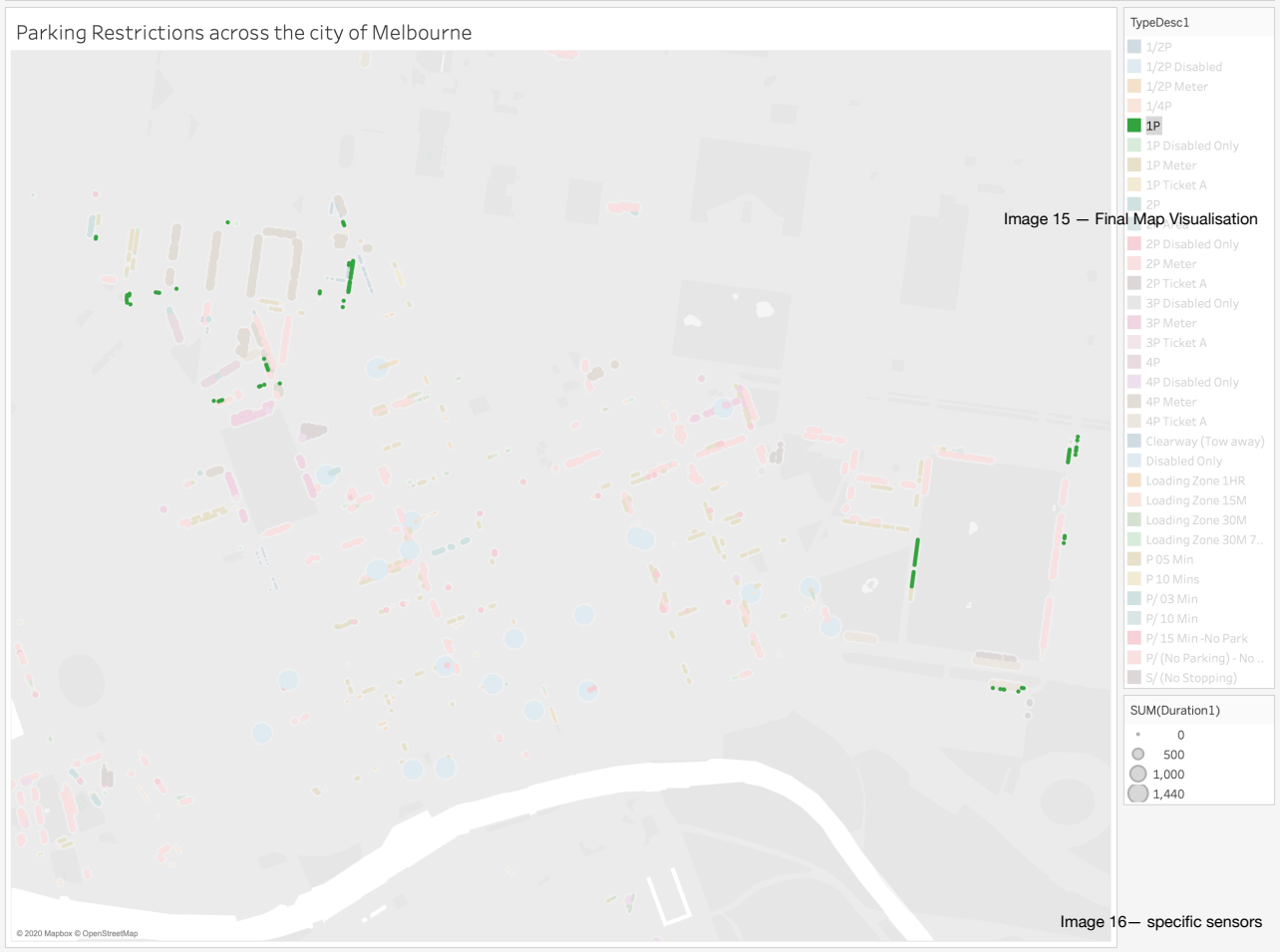


Image 15 — Final Map Visualisation

Image 16— specific sensors

The above like interpretations are discussed more elaborately in the *Conclusion section.*

Up till this point, the wrangling and cleaning has been done using R and visualisations were made using Tableau. Now we try to shape file a map using our third dataset which includes spatial data from ESRI. Unlike individual sensors, this file contains polygons which may help us get a geometric view of the parking areas.
The shape file when imported and plotted in R doesn't look as intuitive as the dynamic dashboards in Tableau but still gives us an idea of how the parking lots (collection of multiple parking bays) may look like from above. It conveys a very systematic non topological view which is significantly faster to read and make on the go changes. The map in Tableau looks more appealing but the one from shape file in R draws a raw skeleton for the parking bays across the city and reveals much more data about the location vectors.

None the less, the shape file *complements* our exploration from Tableau by visualising the geometric progression of parking lots (collection of Parking Bays) around the streets of Melbourne.

Conclusion

The exploration process helped us unveil the trends and purpose of imposing parking restrictions.

At first, we saw that **over 56% of Parking spots were left unoccupied** which means we are coping very well in terms of managing the incoming flow of vehicles.

Australia being second highest country in the world in terms of Cars per Capita[1] made me feel a little suspicious about this conclusion. Again, this can be argued with the fact that ~~*Australia also has big land area and considerably low population*~~[2] when compared with the world but the pace with which Melbourne is attracting businesses and students, and has been growing in terms of population; it is more logical to strike off this reason. Lastly, to aid my claim — I found a study stating that *by the year 2036 we will run out of Car Spaces in Melbourne CBD*[3]. Well this is very contradicting to the judgement drawn from our data. Soon after I realised that 2020 has been an unusual year and Victorian Government has imposed city wide lockdown in the Metropolitan Melbourne *twice*. Even during the weeks without lockdown, people have deliberately avoided going out and the offices have completely transitioned to Work From Home.

You might be confused that I made claim, suspected it and proved it wrong using strong evidences found online. Yet I am in support of what my data says. Yes, considering the fact that COVID is not going away anytime soon and businesses and universities will try to incorporate distance learning as much as they can. So this is the new normal, the lockdown data may have skewed the results but these conditions are here to stay. At the moment the Victorian Government has been managing the use of public space quite well.
Thus the demand for parking bays copes up well with the availability. But how is it made possible? The answer lies within the restrictions imposed by the government on the use of these parking bays.

The restrictions data infers that **majority of meters allow a standard 2 hours of parking time (37%)**, followed by 1 hour (15%), 4 hour (9%) and 3 hour durations (7%). In a nutshell 70% of parking bays are reserved for general public with 1P, 2P, 3P and 4P meters. Apart from that we have certain bays with up-to 24 hours of parking time allowed and these are reserved only for the Disabled drivers. Such reserved spots have been placed in a decently planned manner such that no two *24 hour bays* lie next to each in close proximity. It is ensured that they are available at considerable distances so disabled drivers do not have to suffer. And not having such bays in large numbers ensures that other public is not affected due to reserved spaces as the number of disabled drivers are considerably less. Thus the authorities play fair in maintaining availability and accomodating those with special needs.

Therefore it can be generalised that majority of bays permit parking only up-to 2 hours. This time limit is capped to ensure that people do not unnecessarily exploit the public spaces; especially in busier areas where the duration is reduced to 30, 10 or even 3 minutes just to guarantee that people do not occupy the bay for long periods and give a chance to every visitor to get their work done. The charges and fines due to overuse encourage drivers to make optimal use of their time as well as the bay occupied by them.

As a result, the authorities allow a fair chance (claim 2) to every driver, and thus are able to maintain high availability (claim 1) to meet the demand.

Reflection

The project helped me understand that data wrangling and data cleaning are overlapping domains. Most of the time they are done in parallel. And not matter how much you try to get the data in perfect form, exploration phase reveals that data still needs improvement. None the less, this is a learning that data checking is better when done visually. Despite all the analysis in R, had the data been plotted during the checking phase itself the process could have been smoother.

Coming to the questions proposed a few weeks ago —

1. How does the availability compare with the demand?
2. How do the parking restrictions compare across the city and on different days of the week?
3. What role do restrictions play in ensuring a *fair* chance of finding a spot?

I could answer the first and third queries very specifically and reasonably well but in the second question, I could only answer the first part which compares the restrictions geographically and failed to answer how did the restrictions differ according to days of the week. Reason being, the data is refreshed every two minutes and exporting it gives the real time status of all the parking sensors at the moment of download. I expected the data to be a collection of daily or weekly trends for status of parking spots but this was incorrect on my part. A solution to this could have been fetching the data multiple times, say 24 times a day but this would still not answer how did it change across the week. And downloading it each hour, every day of the week would have required certain scripting and couldn't have been done manually. Again, that could have served me data for one week but I needed a big collection (at-least 4 weeks of data) to make a generalised statement. Thus the second part of second question was a little ambitious considering the sparseness of data. A reasonable tradeoff could have been to compare hourly trends.

Lastly, the use of shape file was not done up-to its capability. I chose this only because we were taught handling shape files in one of the tutorials and I wanted to enhance the degree of difficulty. Halfway though the subject I have understood, that shape files are useful when we need to make frequent changes as they are very fast to read and write but in my project, the data was static and read only once in R and once in Tableau. None the less, it motivated me to revisit the Module 3 and learn about the shape file formats (shp, shx and dbf) and its purpose. Also, at this stage of learning, Tableau seems to be more user friendly than R.

But again, the shape file could have been useful if I were to make spatial changes such as adding new or removing existing parking bays.

Bibliography

1. CarsGuide (2020). *Australian Car Market: Car Sales Statistics & Figures Australia*.
Retrieved from https://www.carsguide.com.au/car-advice/australian-car-market-car-sales-statistics-and-figures-70982

2. Wikipedia (2020). *Demography of Australia.*
Retrieved from https://en.wikipedia.org/wiki/Demography_of_Australia

3. ABC News (2018). '*Major Changes' to Melbourne CBD car usage needed, warns council parking report.*
Retrieved from
https://www.abc.net.au/news/2018-06-21/melbourne-city-council-considers-sweeping-changes-to-push-cars/9892998

4. OpenData (2020). *City of Melbourne.* Retrieved from
https://data.melbourne.vic.gov.au/Transport/On-street-Parking-Bay-Sensors/vh2v-4nfs?_ga=2.61374956.1505471538.1600169540-328008717.1600169540

Street Parking across the city of Melbourne

Section 2 — Visualisation

Name -	Simran Singh Gulati
Student ID -	31125301
Lab Number -	Online 03 - Wednesday, 6-8 pm
Tutor Name -	Jie (Lewis) Liu, Jeffery Chieh Liu

Introduction

For a quick background from the exploration phase, we surveyed the *On Street Parking* data for the city of Melbourne. Our analysis helped in unwinding the critical aspects associated with a parking meter, primarily aiming at demand for parking spots.

So I'll try to compare the availability of the parking bays along with the restrictions imposed on them, and the variation in trends across different parts of the city. Revisiting the question proposed in Week 3, here's a quick glimpse on the areas of concern:

1. How does the availability compare with the demand?
2. How do the parking restrictions compare across the city and on different days of the week?
3. What role do restrictions play in ensuring a *fair* chance of finding a spot?

Of the listed questions, I'll be answering all three except the later half of the question two which includes weekly trends. For this particular question I'll be limiting to the area comparison but the other two would still stay the same.

The audience for my visualisation is not limited to my classmates but may also be helpful for the residents of Melbourne or the general public. More importantly, the exploration results through a visualisation can be utilised by the government officials for better strategising their expansion and/or modifying the current parking rules.

Design

Following the Five Design Sheet methodology, I started by brainstorming visualisation ideas but soon enough I realised that I should first highlight the three major domains inferred from my questions. Thus I listed *availability*, *area* and *duration* as my areas of concern. As evident from the first sheet (full size attached in the appendix), I scribbled the possible visualisations for each of the three.

For comparing *occupied versus the unoccupied* bays I considered a pie-chart, column chart, bar graph, stacked bar graph and a spatial representation wherein I intended to show the two types of parking bays on a map, each in a different colour.

To show distribution across the city I proposed the use of a heat map or a bubble chart with each bubble representing a suburb and its radius proportionate to the number of parking bays in that suburb with colour coding to show the availability status.

Finally for comparing the parking restrictions with prime focus on the permitted duration, I listed pie-chart, line-chart and a spatial visualisation wherein each parking bay would be a circle on the map and its radius is proportionate to the duration permitted in that area.

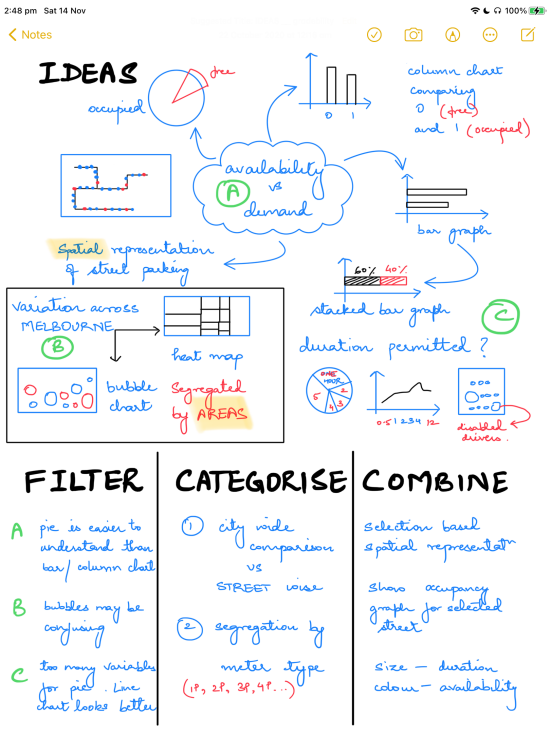


Image 1 - First Design Sheet

On analysing the offerings, I narrowed down the designs by removing the least appealing visualisations gradually. For instance - while the pie chart is great for comparing the occupancy (free or not) it would be a disastrous idea to compare the duration types as there would be a lot of sections on the same chart. Similarly I chose to drop the bubble chart for a heat map. Lastly, I chose a map approach to show variations across the city.

Walking on these lines, I shall briefly discuss the three shortlisted dashboards:

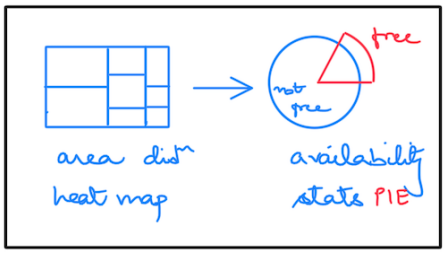


Image 2 - Design 1

This image on the left refers to the first design wherein a heat-map is used to represent parking availability in an area.

The denser areas (bigger and darker) have greater number of parking bays. Each rectangle is clickable and reveals parking restrictions on mouse hover.

Besides, we have a pie-chart showing availability of parking bays. By default the pie shows statistics for the entire city but clicking on an area in the heat map restricts the pie to show occupancy in that particular area.

Although we touched each aspect in the last design, I felt using a map would be a better alternative to a heat map, it may help engage the user more effectively.

So the webpage includes a stacked bar chart for the occupancy stats for the city. And clicking on the category (occupied or not) shows those bays on the map to its right.

Say a user clicks on occupied parking bays (red) on the stacked bar chart, the map then highlights the occupied spots. And rest of the parking bays (free) fade out in colour (blue to light blue).

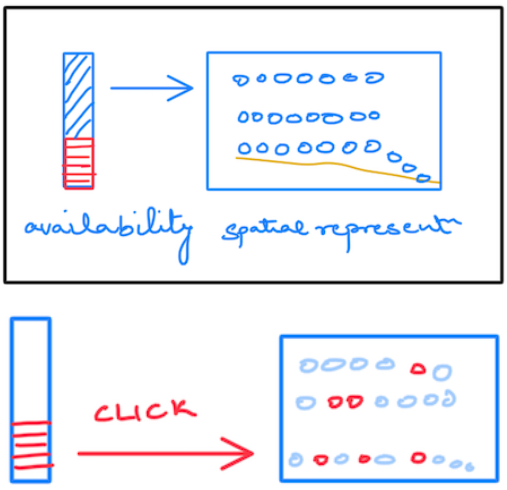
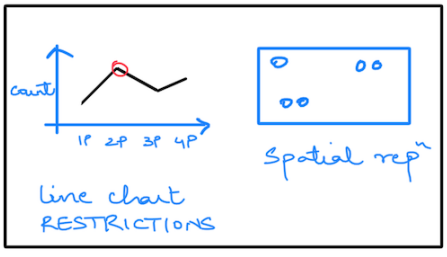


Image 3 - Design 2

First design focusses glorifies the area segregation by following a street first approach. On the other hand, second design acts more like a bird's eye view of the city. Though both of them encompass the targeted questions, the restrictions data is not utilised to its full potential. Having meter information on hover (in first design) is low-key limiting its use case.

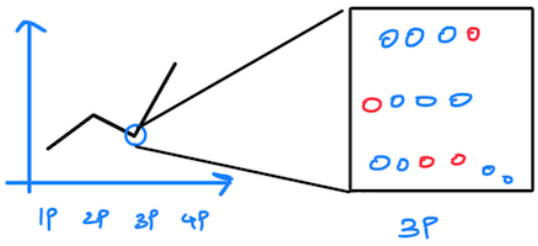
Image 4 - Design 3



Thus for the final design, I started with focus on meter types.

So there's a line chart showing the total number of bays under each meter type. And clicking on a meter type shows all the parking bays in the city monitored under that specific meter.

Image 5 - Design 3



On scrutinising its functionality, I felt the the comparison for availability may be influenced in a map view.

Although colours can be used to get an overview but scaling/zooming on map may target one colour to over power the other.

Similar problems may arise when a lot of parking bays are concentrated in a particular area. It may cause overlapping on zooming out. And this hampers the comparison for availability. All in all, availability can be better visualised with a pie or bar chart. At the same time having a spatial component will make the design more engaging for the user.

In hindsight, I decided to go with first visualisation with an addition of the city map from design number two and replacing heat-map with a bubble-chart as per the feedback.

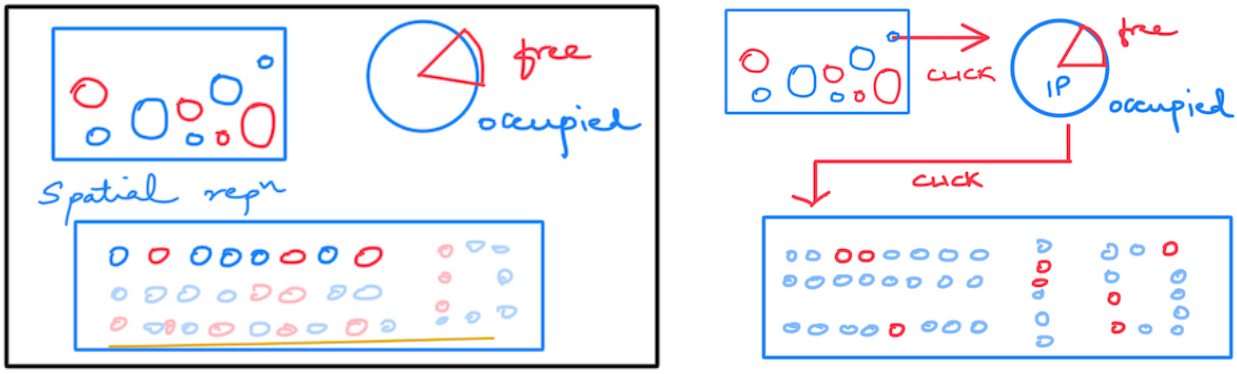


Image 6 - Final Design

As can be seen in the picture above, my final design (updated as per feedback from presentation) has a total of three visualisations, each daisy chained with one another to support interactivity.

At first the user is greeted with a bubble chart on the top left. Each circle represents a specific meter type and hence accounts for the parking restrictions in an area. This would be beneficial to draw *most common* permitted durations for street parking. Busier areas like Flinders are expected to have lower permitted duration to ensure regular turn over. At the same time there must be higher number of bays for disabled drivers. And in the *not as busy* suburbs we expect higher duration times and lesser number of accessibility or VIP bays.

Although the meter names are self explanatory (2P - 2 Hour Parking, 1P - 1 Hour Parking, etc.), it would be hard for a first time user to decode the naming convention. Thus, to avoid any inconvenience, we must add the functionality to reveal such associated information via a mouse hover.

Next up we have a pie chart showing the availability stats for the entire city. If the user wants to check the demand for a particular meter type say for 2P meters, he can simply click on a meter from the bubble chart and the the pie dynamically restricts the comparison for occupancy for the selected meter type. We add another layer of functionality by making the pie attributes clickable which can be better understood if discussed in parallel to the third visualisation.

As pointed out earlier neither just the numbers (pie/bar chart) nor just the map is sufficient, we must use them along side each other for best narration. Thus a map of the city of Melbourne is added. It shows each parking bay as a circle. Plus we use two contrasting colours to show the availability status of each bay. An example could be *green* for available/free and *red* for occupied/un-available bay.

Most importantly, the user can select the status from the pie chart and only those bays would be on focus in the map. For example, if I click the unavailable (red) parking bays on the pie chart then the available (green) ones fade out on the map. So the user has control to see either all the bays or a specific type of bays. Both the pie chart and map are supported with a reset button to change to default view.

In nutshell, at the beginning, user sees a bubble chart of meter types, a pie chart for availability and a spatial representation of all the bays on a map. They may choose to restrict the availability stats (pie-chart) for an area selected from the heat map. Furthermore, they may choose to display either of the two types (free or not) exclusively on the map.

While the bubble chart and pie-chart cater to generalisation and give answers at a glance, the map not only makes it appealing to look at but also allows better association of those two. And makes it easier for the user to explore underlying insights.

Now that we have discussed the raw structure of our final design, let's make it more intuitive by styling it to cater our perceptual system. For the bubble chart there's a total of 33 meter types, so we aggregate those with the same permitted duration under one colour to reduce ambiguity. At a broader scope we can see 5 categories each encompassing 5 to 6 individual meters. Each category is visualised as a bubble with contrasting (diverging) colours derived from the Brewer palette, making it easier for the human brain to draw results.

Similarly the pie chart consists of 2 heads, namely *Occupied* and *Unoccupied.* We want to convey the message of opposing attributes. The sensory system of humans is bound to perceive red colour as alerting, for instance in notifications prompts and traffic lights. And green colour subconsciously hints progressive or acceptable incidents. Therefore we use the same ideology in designing the pie chart with a shade of red for unavailable bays and green for free bays. I say shades of red and green, and not pure colours to make the webpage look appealing. I feel the use of base colours is very punchy to the eyes which overpowers other visualisations. So I will stick with subtle shades to blend in with surrounding figures.

Finally coming to the spatial segment, there's over a four thousand sensors across the city. Deploying them altogether as markers not only brings down the performance of our application but is also overwhelming for the reader. We will approach this issue in two steps, firstly we try to use clusters based on proximity and secondly we replace location markers (default icon) with circles. Clustering the sensors helps get a count of points in a street which would be very hard otherwise. More importantly it makes it easier to zoom in and out on application map which otherwise would lag heavily (irrespective of the machine's compute capability).

Lastly, to make the city map easier to interpret we use colour coding for defining its availability exactly like we did in pie chart. This makes the transition among the visualisations easier for the user. Each bay shows its ID on being clicked plus we add the hover functionality to retrieve the meter description.

Implementation

In this phase we turn the proposed designs into a workable client ready application.

We must note that data at this stage is in raw form and needs restructuring to be fed into application frameworks. Having gone through the exploration phase in the last project, we now are focussing only on the primary restrictions.

For a quick recap, each bay is associated with a bunch of rules varying with time and day of the week but due to sparseness constraints (discussed in detail in exploration project) we chose to restrict to main restrictions for each bay. It refers to the rules applied on a bay for majority of times.

We begin by importing four libraries, each with a specific purpose -

- dplyr
- ggplot
- leaflet
- shiny

Wherein dplyr is used for data wrangling and transformation purposes; ggplot helps in creating bubble and pie charts; leaflet allowing map functionality and lastly shiny to integrate these components in to a full fledged application.

While most of the functions can be performed using base functions available in R, each of the above mentioned libraries helps in refining those codes. For instance dplyr allows piping component and ggplot helps create graphs that aesthetic than the ones that come built in. And leaflet reduces the developer's hassle by miles! It provides seamless integrating with a minimalistic script. Complex features like labels, icons, popup etc. can be altered seamlessly in leaflet. Finally, shiny allows plug and play to create a web application which is a lot easier than JavaScript as taught in the lectures. Migrating visualisation created with ggplot into a shiny framework is as easy as printing "hello world" and is much recommended over D3 unless the developer wants granular control which is not in our case.

As pointed out earlier the data is spread among 3 different datasets and needs to be combined. Thus we import electronically generated sensors data and human typed parking restrictions into two separate data-frames. Next up we drop unnecessary columns from the restrictions file and merge them into one data frame using an inner join. The two tables are joined using the unique bay_id for each parking spot. As each visualisation has separate requirements and areas of focus, we extract elements specific to each visualisation in 3 separate tables.

For the bubble chart, we pick the column TypeDesc1 from the fused dataset and convert it into table that gives us the count of each meter-type. Then we add two more columns, in first I assigned a serial number for each row and in the second I combined meters with same permitted duration into categories. For instance, there were separate entries for 1P 1P Disabled and 1P Ticket A, so I added into a generalised section 1P.

	MeterNo	Freq	Meter
1/2P	1	69	1/2P
1/2P Disabled	2	1	1/2P
1/2P Meter	3	49	1/2P
1/4P	4	81	1/2P
1P	5	117	1P
1P Disabled Only	6	4	1P
1P Meter	7	509	1P
1P Ticket A	8	8	1P
2P	9	58	2P
2P Area	10	1	2P
2P Disabled Only	11	47	2P
2P Meter	12	1219	2P
2P Ticket A	13	169	3P
3P Disabled Only	14	4	3P
3P Meter	15	207	3P
3P Ticket A	16	20	3P
4P	17	59	4P
4P Disabled Only	18	22	4P
4P Meter	19	304	4P
4P Ticket A	20	101	4P
Clearway (Tow away)	21	20	Special
Disabled Only	22	25	Special
Loading Zone 15M	23	74	Special
Loading Zone 1HR	24	9	Special
Loading Zone 30M	25	61	Special
Loading Zone 30M 722	26	41	Special
P 05 Min	27	5	Special
P 10 Mins	28	17	Special
P/ (No Parking) - No Park	29	2	Special
P/ 03 Min	30	1	Special
P/ 10 Min	31	11	Special
P/ 15 Min -No Park	32	5	Special
S/ (No Stopping)	33	2	Special

Image 7 - Data after processing for Bubble Chart

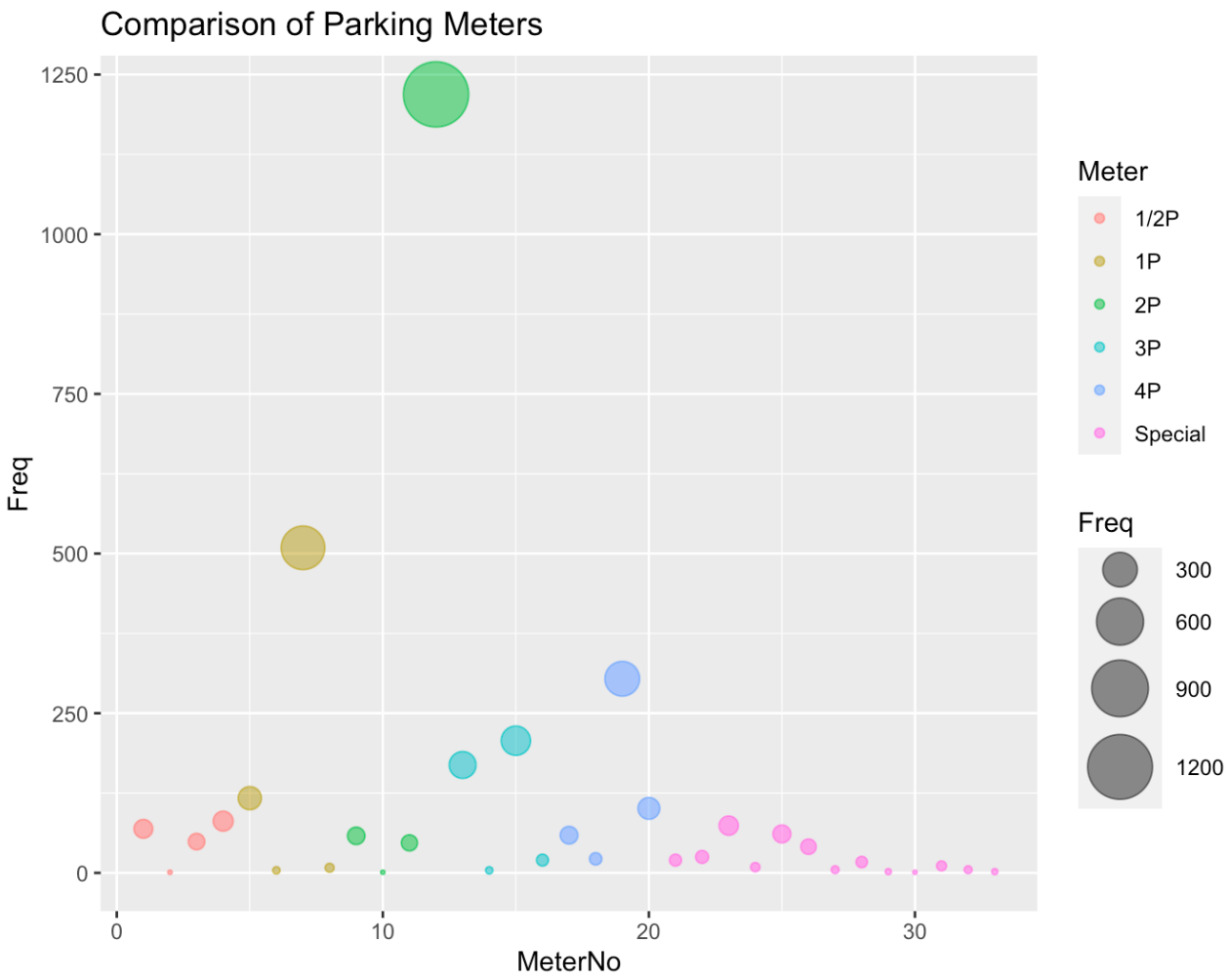
Unlike automating the data imputation as for the bubble chart, I simply the printed the data for pie chart using the table function on the *status* attribute. And manually created a dataframe for the associated values. As there were only four values, it was faster to do it manually plus there's a lot less chance of error as there are just 2 rows and 2 columns.

```
      Status Count
1   Occupied  1453
2 Unoccupied  1869
```

Image 8 - Data created for Pie Chart

Now that we have required data, we start plotting the visualisations.

Image 9 - Bubble Chart



The picture above represents the bubble chart created for our dashboard. And below can be found the code for the same :

Image 10 - Code for Bubble Chart

```
# create vis 1 of 3 - bubble chart
my_bubble <- ggplot(bubble_data, aes(x = MeterNo, y = Freq)) +
  geom_point(aes(color = Meter, size = Freq), alpha = 0.5) +
  scale_size(range = c(0.5, 12)) + ggtitle("Comparison of Parking Meters")
```

Similarly we plot the pie chart directly from the table created specifically of each graph:

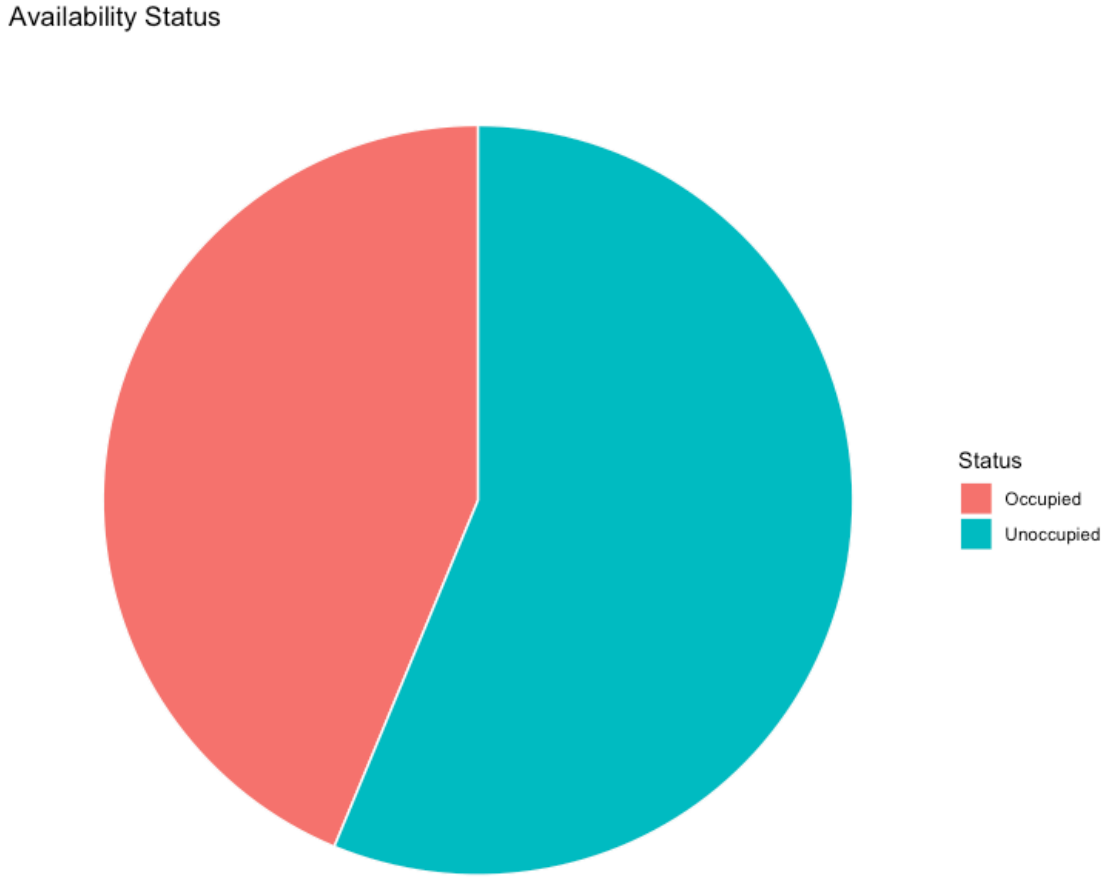


Image 11 - Pie Chart

```
my_pie <- ggplot(pie_data, aes(x="", y=Count, fill=Status)) +
  geom_bar(stat="identity", width=1, color="white") +
  coord_polar("y", start=0) + theme_void() + ggtitle("Availability Status")
```

Image 12 - Code for Pie Chart

Lastly we look at the map created from leaflet :

```
my_map <- leaflet(data = df) %>% addTiles() %>% addCircleMarkers(
  ~lon, ~lat,
  popup = ~bay_id,
  label = ~TypeDesc1,
  color = ~pal(status),
  stroke = FALSE, fillOpacity = 0.5,
  clusterOptions = markerClusterOptions())
```

Image 13 - Code for City Map

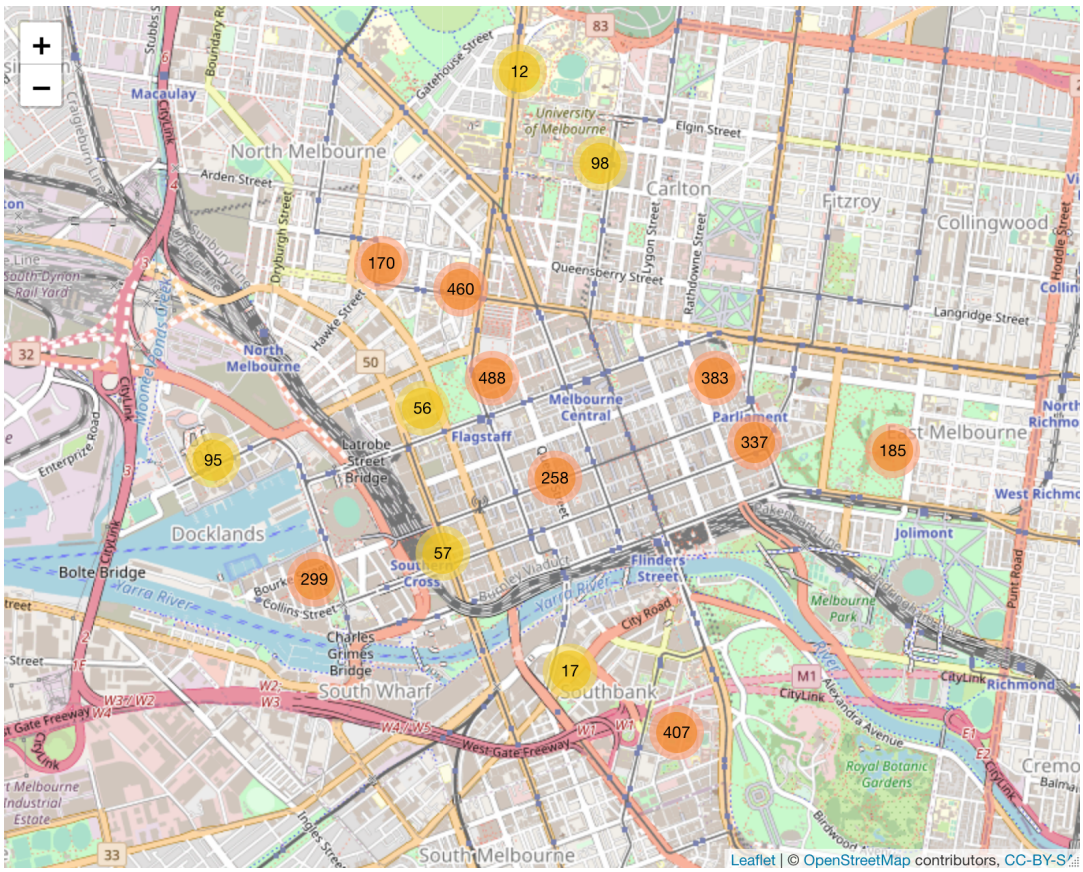


Image 14 - City Map with Parking Clusters

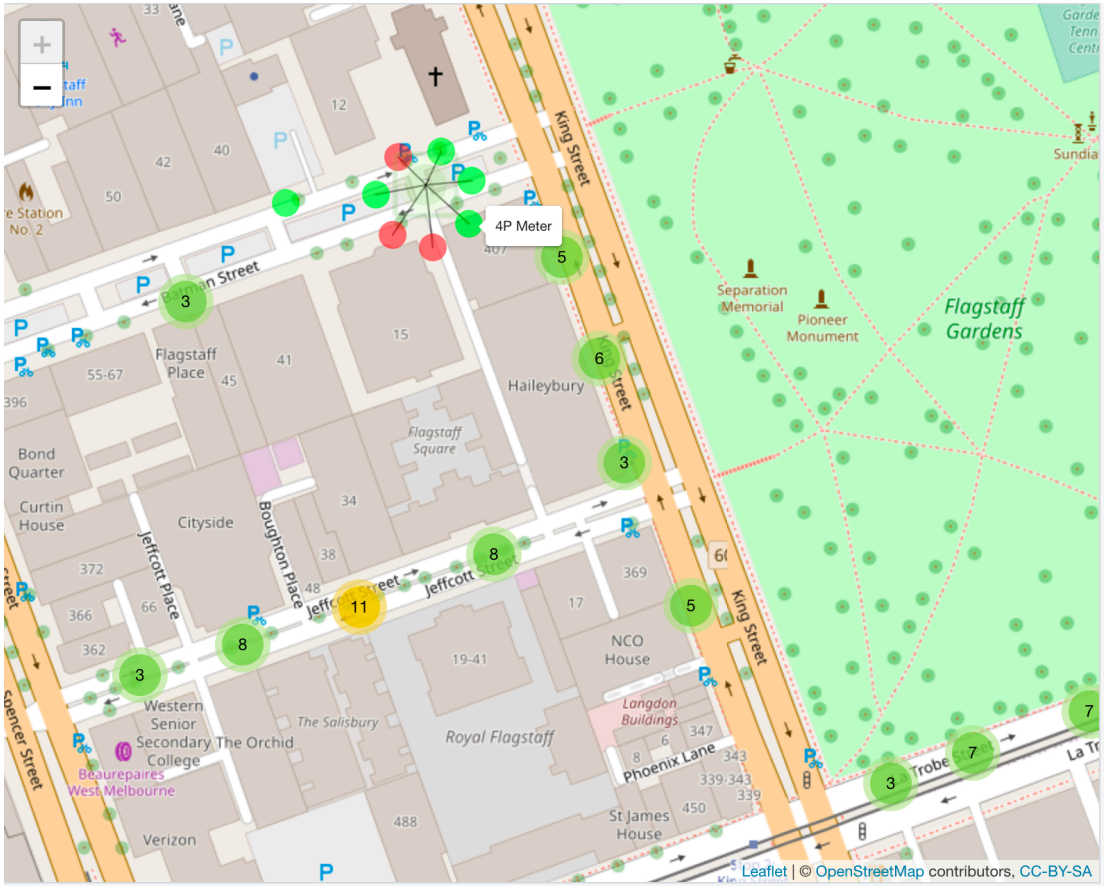


Image 15 - City Map with Zoom and unclustered Parking Lot (labelled)

Looking back at what we have done till now :

1. Imported data from multiple sources
2. Restructured and fused them into 1 data-frame
3. Extracted data for each visualisation
4. Generated visualisations separately

And now is the final step wherein we combine all the visualisations in an application or graphic enabled dashboard via the use of shiny in R. The reason we did all the data processing prior to feeding it in the shiny framework, is that it reduces run time and improves app performance of the application.

The front end design was stored in a variable *ui* and the backend logic was stored in the variable *cache* and they were run together in the function shinyApp.

```
ui <- fluidPage(
  # page heading
  headerPanel("Surveying Street Parking across the city of Melbourne"),
  # heading 1 of 2
  h3("Statistical Results"),
  # plot 2 graphs in 1 row
  plotOutput("my_graph"),
  # heading 2 of 2
  h3("Spatial Representation"),
  # plot city map
  leafletOutput("my_map")
)

# back end
server <- function(input, output, session) {
  output$my_graph <- renderPlot({
    # to generate 2 separate visualisations in 1 row
    gridExtra::grid.arrange(
      # vis 1 - bubble chart
      my_bubble,
      # vis 2 - pie chart
      my_pie,
      nrow = 1
      )
    })
  # vis 3 - map
  output$my_map <- renderLeaflet({
    my_map
  })
}

# run app
shinyApp(ui, server)
```

Image 16 - Code for Shiny Integration

User Guide

The final dashboard looks like:

Image 17 - Application Screenshot - Final Dashboard

Wherein the user can select a meter (bubble) from bubble chart that changes the pie chart and may select the occupancy status from the pie chart. You may zoom in/out on the map using scroll functionality or the buttons (plus/minus) on the top left inside the map.

The image on the right is a scaled in view of the map wherein the colour of the cluster shows the density of parking bays in that area and the number in its label shows the exact count of those bays.

Red implies large number of bays, yellow represents moderate bay count and green is the highest level of zoom with the lowest count in a cluster.

It can be clicked to unveil individual parking bays as in the next image.

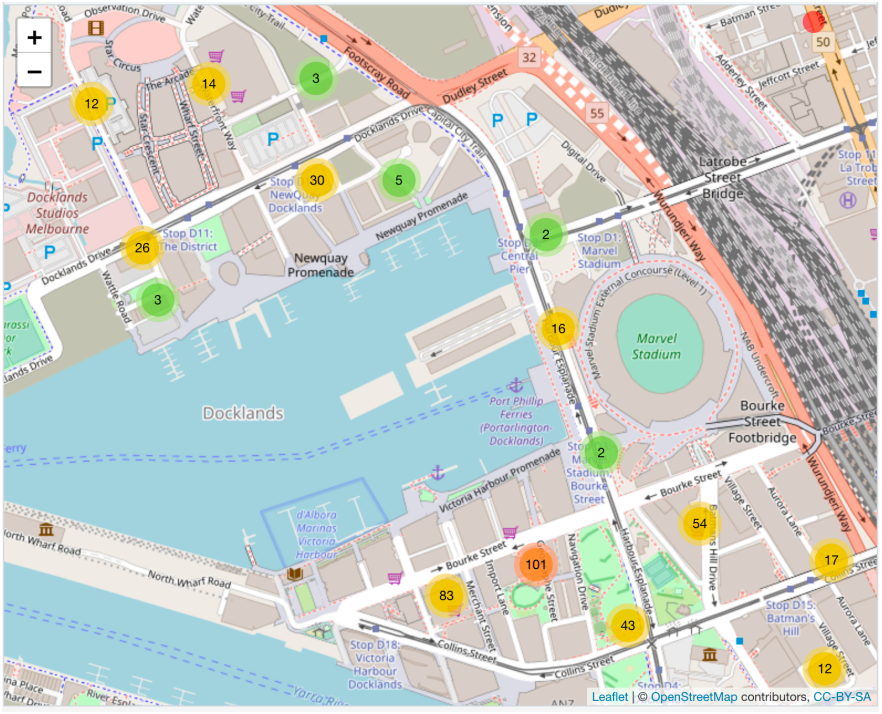


Image 18 - Zoomed In View of the City Map

As evident from the plus button having turned grey in colour, the map can not be zoomed in any further. Plus the clusters turned green also validate the click-ability and hence no scope for zooming in.

On further clicking a cluster, it reveals the all the bays monitored under that meter. And the name of the meter can be retrieved by hovering over any parking bay (not the cluster) as can be seen in the image above.

Note : the clusters have a frequency count on them as a label, while underlying bays have the meter type as the label.

Most importantly, the bays have separate colour encodings for the availability status. The spots which are vacant/available are green in colour and those occupied already and not available are red in colour.

Conclusion

Beginning with design phase, the bubble chart could have been replaced with some animations like a *motion chart* from Google Visualisations. At current stage it is quite static and the motion chart would have explained the variation in meter types over a period of time. Current results are an enhancement of a basic plot like line chart, so the bubble chart is not utilised to its full potential.

In the coding phase, while combining the meter types in a new column, use of Regular Expression could have reduced the chance of error. Having achieved that manually, I had to make last minute changes wherein both “1/2P” and “1P” were incorrectly put under “1P” due to careless reading. Such human errors in data entry must preferably be handled via computer logic.

Earlier all the data processing was done directly in server segment of Shiny, which made the application load a little late. Having dedicated data frames for each visualisation (created before hand) passed into the server saved the run time. And the application loaded as soon as the button was clicked.

Similarly, while plotting the location coordinates on map, I experienced huge lag while moving around the on the map. This was due to the large number of rows that had to be plotted. Using *clusters* solved this problem and it runs pretty smooth now. Plus it allowed me to use circle markers instead of location markers. The location pin points were overlapping and not very good at explanation so the use of translucent circles helped in that as well.

Graphing and Interactivity is a lot easier in Shiny than the D3 library of JavaScript. Considering D3 provides high level of customisation, it may be better for UX/UI design but for the scope of our project Shiny seems to make the process a lot simpler and fun.

Overall, I felt the data exploration is statical and more important for Machine Learning. For the data visualisation part, I felt basic plots are very helpful for understanding the insights or trends but design process is a whole new field to be explored. Firstly, a sophisticated application demands development knowledge like HTML/CSS, and JS/Shiny. Next it encompasses a lot of design methodologies to hack someone’s psychology or thinking process to grab their attention. And understanding what audience an application is targeted for.

For a perfect application, we need domain knowledge of 3 separate areas — statistical analysis, application development and UX/UI design. So data visualisation seems like a very intense specialisation which has a lot to explore. Though diverse and hard, the visualisations discussed in hall of fame each week make me think that it has promising future alongside the data lifecycle.

In nutshell, it is a multidisciplinary field and helps convey the complex numbers/patterns to a layman very easily.

Bibliography

1. OpenData (2020). *City of Melbourne.* Retrieved from https://data.melbourne.vic.gov.au/Transport/On-street-Parking-Bay-Sensors/vh2v-4nfs?_ga=2.61374956.1505471538.1600169540-328008717.1600169540

2. VicRoads(2020). *Parking: Vic Roads.* Retrieved from https://www.vicroads.vic.gov.au/safety-and-road-rules/road-rules/a-to-z-of-road-rules/parking

3. City of Melbourne(2020). *Parking Rules.* Retrieved from https://www.melbourne.vic.gov.au/parking-and-transport/parking/parking-rules/Pages/parking-rules.aspx

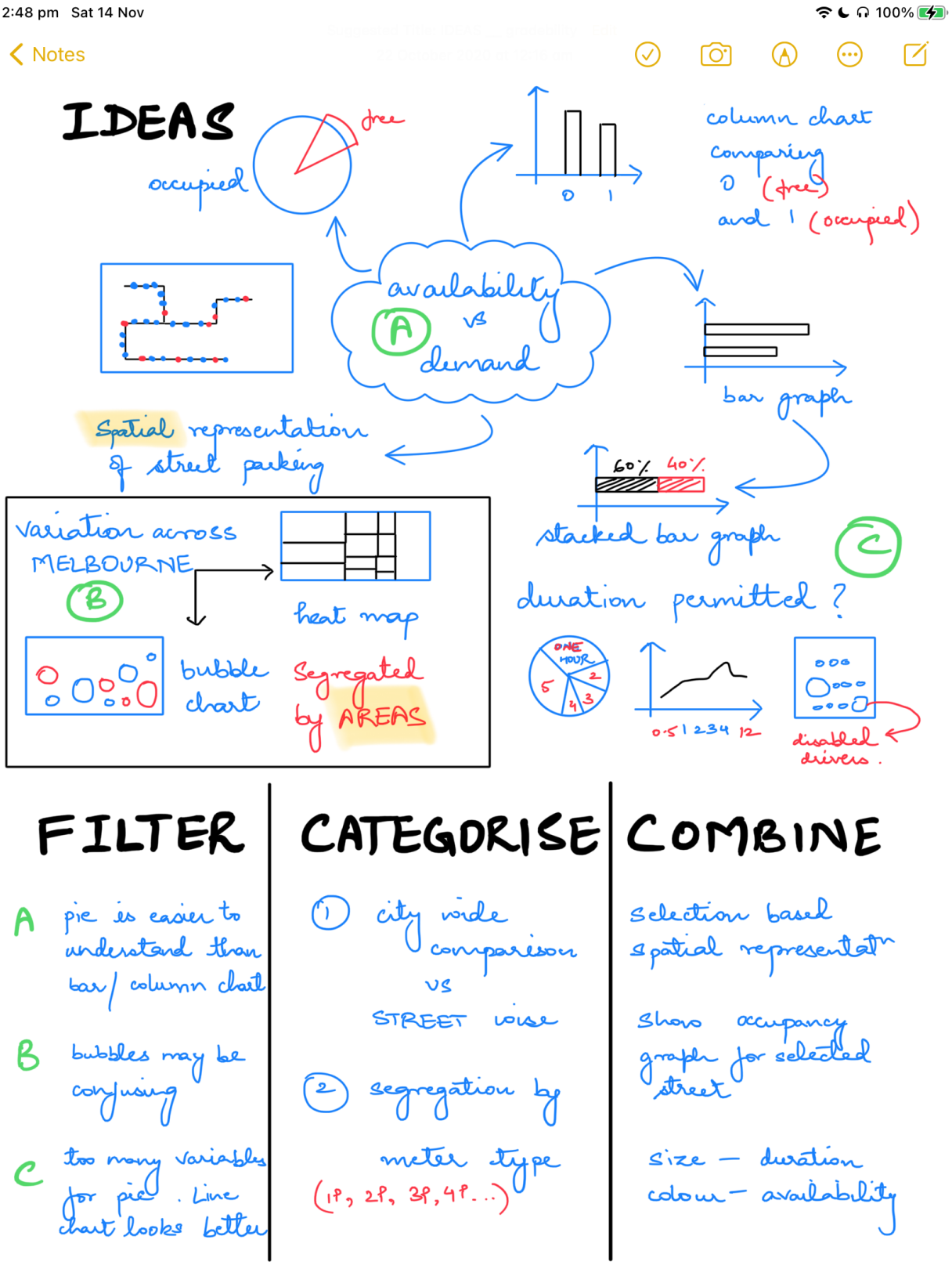
4. Wikipedia (2020). *Demography of Australia.* Retrieved from https://en.wikipedia.org/wiki/Demography_of_Australia

5. CarsGuide (2020). *Australian Car Market: Car Sales Statistics & Figures Australia*. Retrieved from https://www.carsguide.com.au/car-advice/australian-car-market-car-sales-statistics-and-figures-70982

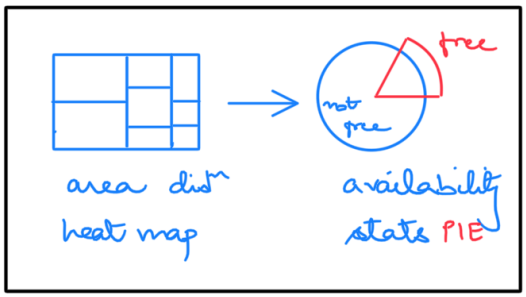
6. ABC News (2018). '*Major Changes' to Melbourne CBD car usage needed, warns council parking report* Retrieved from https://www.abc.net.au/news/2018-06-21/melbourne-city-council-considers-sweeping-changes-to-push-cars/9892998

Appendix

Five Design Sheet :

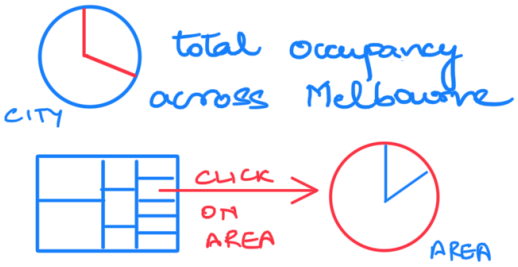


LAYOUT



TITLE Comparing parking across streets of Melbourne
AUTHOR Simran Singh
DATE 23/10/2020
SHEET # 2

FOCUS



mouse click on heat map restricts pie chart to that particular street (selected)

OPERATIONS

① show what are the most populous areas

i.e. streets with most no. of parking bays.

→ a heat map comparing streets with parking across the city of Melbourne

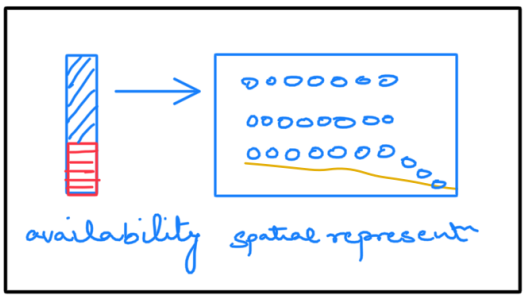
② display a comparison of availability of parking spots

by default it shows total no. of occupied spots vs free spots

DISCUSION

+ covers area
+ shows availability
− parking restrictions not included

LAYOUT



TITLE Analysing car parking across streets of Melbourne
AUTHOR Simran Singh
DATE 23/10/2020
SHEET # 3

FOCUS

selecting unoccupied spots on bar graph will only show such spots on map. while already occupied bays fade out.

OPERATIONS

① The stacked bar graph shows what proportion of parking spots are occupied or free

② A map of city is placed to the right of the bar graph

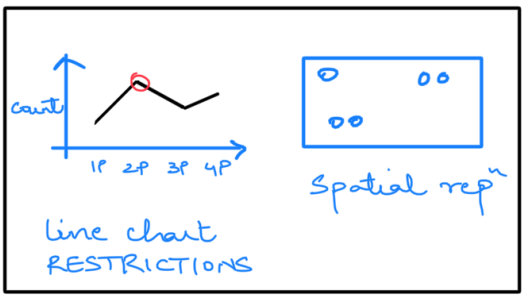
The map shows the parking locations across the city of Melbourne.

And shows only 1 category at a time eg only occupied parking bays

DISCUSION

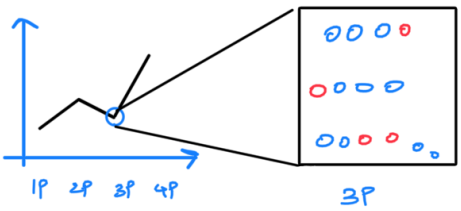
+ bird's eye view of city
− lacks region specific functionality
− parking meter info not used.

LAYOUT



TITLE Role of parking restrictions in availability
AUTHOR Simran Singh
DATE 23/10/2020
SHEET # 4

FOCUS



clicking on the meter type shows parking bays monitored under that meter.

OPERATIONS

① user is greeted with comparison of meter types

each meter type denotes a set of restrictions focussed around permitted duration in that area

② a map is attached to this line chart

the map shows bays associated with only 1 meter type at a time.

ex- all spots monitored by 1P meter.

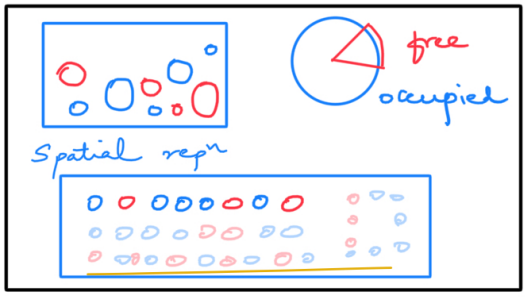
DISCUSION

+ accounts the role of restrictions

+ incorporates availability and region

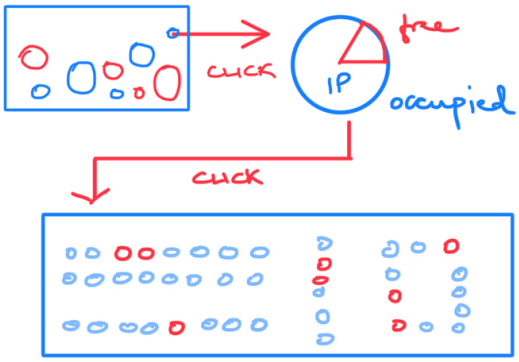
− comparisons may be influenced in map view.

LAYOUT



TITLE Role of parking restrictions in availability
AUTHOR Simran Singh
DATE 23/10/2020
SHEET # 5

FOCUS



- user selects meter type
- pie chart shows meter stats
- map highlights meter type

(OPTIONAL) — user selects from pie char — map only retains selection

OPERATIONS

① bubble chart — regulated by parking durat^n
meter type ≡ duration

② pie chart — availability comparison for selected meter type

default — city wide comparison

③ map — parking bays under the selected meter are highlighted
rest of the bays fade out

DETAILS

req.s

time to build — 10 days
estimated delivery
— 16 Nov, 2020

Language — R (wrangling + dev)
libraries — dplyr, ggplot, leaflet and Shiny
Software — R Studio & Chrome.

Shapefile plot

FIT5147 S2 2020 Data Exploration Feedback

Student Name: Simran Singh Gulati (sgul0007)
Marker: Lewis

	Outstanding (75	Adequate(60-75	Not Adequate ((	Comments
Data Checking and Wrangling				
Appropriate checking for errors in data	X			Your error checking is done properly with sufficient details provided. However, have you checked and found any outliers or abnormal values?
Appropriate cleaning and reformatting	X			It is very good that you have provided a summary of all your datasets with links provided. Your data wrangling is done properly with sufficient details provided. Value removals and sensible with justifications provided. Well done!
Managed to get data into R or Tableau	X			
Data Exploration				
Completeness/thoroughness	X			Your proposed questions should be listed when you are doing data exploration (Yes, you have provided subsections, one for each question, but you could have made it clearer).
Use of appropriate visualisations and/or analytical measures	X			Legends should be provided for image 12. It is probably not a good idea to use pie charts when you have many restrictions types. Similar problem for image 15. Too many colours and too many types in the image. It is difficult to read and to find which type is where. However, looking at image 16, it starts to make more sense when selection is enabled. It would be nice if you can overlay image 17 or draw it directly on top of a map. Even lat and lng informations are provided, it is still difficult to recognise the locations and places from the image.
Degree of Difficulty				
including, but not limited to, the use of non-tabular data, significant wrangling or cleaning required, large dataset, multiple data sets.		X		You have many datasets to explore, some of which are large in size. As mentioned above, your data wrangling is done nicely with sufficient details provided. I liked the way you conduct the analysis using a story-telling like style. Not only it makes the report much fun to read, but also it makes it easy for me to follow and maks sense of your analysis and discussions. The conclusion is informative with take-away messages properly summarised.
Written Report				
Quality of writing, referencing, images, logical structure (follows suggested format)	X			Overall, Overall, the report is of good quality.
Completeness: The report contain the following sections: Introduction/ Data Wrangling/ Data Checking/ Data Exploration/ Conclusion/ Reflection/ Bibliography	X			You should have clearly listed your questions in both introduction and exploration sections. You have included all required sections in your report. It is good that you have provided some references.
Submitted on time				
Late Penalty (5% of total mark [33%] per day)	X			