PROJECT REPORT

SIGNS WITH SMART CONNECTIVITY FOR ROAD SAFETY

TEAM ID : PNT2022TMID12612

KARPAGAM COLLEGE OF ENGINEERING

TEAM MEMBERS:
SRIHARISH PK
SURYA S
SURIYA PRAKASH S
PRADEEP G

INDEX

CHAPTER 1

1. INTRODUCTION

1.1 Project Overview

▪ To replace the static signboards, smart connected signboards are used.

▪ These smart connected sign boards get the speed limitations from a web app using weather API and update automatically.

▪ Based on the weather changes the speed may increase or decrease.

▪ Based on the traffic and fatal situations the diversion signs are displayed.

▪ Guide (Schools), Warning and Service (Hospitals, Restaurants) signs are also displayed accordingly.

▪ Different modes of operations can be selected with the help of buttons.

1.2 Purpose

▪ Smart Traffic Management is a system to monitor and control traffic signals using sensors to regulate the flow of traffic and to avoid congestion for a smooth flow of traffic.

▪ Prioritizing traffic like ambulances, police etc. is also one application comes under smart traffic management..

CHAPTER 2

2. LITERATURE SURVEY

2.1 Existing problem

▪Analysis of crash data has suggested a link between roadside advertising signs and safety.

▪ Research suggests that crash risk increases by approximately 25–29% in the presence of digital roadside advertising signs compared to control areas.

▪ On the other hand, static roadside advertising signs have not been linked with differences in the crash count.

▪ However, this finding is contrary to previous research thatsuggests differences in crash counts exist in the presence of static roadside advertising.

▪ The quantity and quality of available evidence limit our conclusion.

▪ Fixed object, side swipe and rear end crashes are the most common types of crashes in the presence of roadside advertising signs.

▪ In addition, drivers showed increased eye fixations and increased drifting between lanes on the road.

2.2 References

CASE STUDY I:

Topic: Design and Evaluation of an Adaptive Traffic Signal Control System.

Author: Rongrong Tian, Xu Zhang .

YEAR:2017

Abstract: The TRANSYT traffic modelling software to find the optimal fixed-time signal plan and VISSIM micro-simulation software to affirm and evaluate the TRANSYT model and to help assess the optimal signal plan; build an adaptive frame signal plan and refined and evaluated the plan using VISSIM with VS-PLUS emulator. Through micro-simulation, it was shown that delay in the signal control was shortened noticeably than that in the fixed time control.

CASE STUDY II:

Topic: Road safety analysis using multi criteria approach: A case study in India

Author: Shalini Kanugantietal.

YEAR:2017.

Abstract: The TRANSYT traffic modelling software to find the optimal fixed-time signal plan and VISSIM micro-simulation software to affirm and evaluate the TRANSYT model and to help assess the optimal signal plan; build an adaptive frame signal plan and refined and evaluated the plan using VISSIM with VS-PLUS emulator. Through micro-simulation, it was shown that delay in the signal control was shortened noticeably than that in the fixed time control.

CASE STUDY III:

Topic:A New Genetic Algorithm Based Lane-By-Pass Approach for Smooth Traffic Flow on Road Networks

Author: Shailendra Tahilyani.

YEAR:2012.

Abstract: To developed a new lane bypass algorithm for route diversion given a result in smooth traffic flow on the urban road network. Genetic algorithms are utilized for the parameter optimization. replace existed traffic signals with a system that are monitored the traffic flow automatically in traffic signal and sensors are fixed in which so the time feed are made dynamic and automatic by processed the live detection

2.3 Problem Solution Definition

CHAPTER 3

3. IDEATION AND PROPOSED SOLUTION

3.1 Empathy Map Canvas.

3.2 IDEATION AND BRAINSTORMING

Step1:TeamGathering,CollaborationandSelecttheProblemStatement:

Conducting a brainstorm

Executing a brainstorm isn't unique; holding a productive brainstorm is. Great brainstorms are ones that set the stage for fresh and generative thinking through simple guidelines and an open and collaborative environment. Use this when you're just kicking-off a new project and want to hit the ground running with big ideas that will move your team forward.

15 minutes to prepare
30-60 minutes to collaborate
3-8 people recommended

Created in partnership with

Before you collaborate

A little bit of preparation goes a long way with this session. Here's what you need to do to get going.

15 minutes

A **Choose your best "How Might We" Questions**
Create 5 HMW statements before the activity to propose them to the team.

B **Set the stage for creativity and inclusivity**
Go over the brainstorming rules and keep them in front of your team while brainstorming to encourage collaboration, optimism, and creativity.

1. **Encourage wild ideas** (If none of the ideas sound a bit ridiculous, then you are filtering yourself too much.)
2. **Defer judgement** (This can be as direct as harsh words or as subtle as a condescending tone or talking over one another.)
3. **Build on the ideas of others** ("I want to build on that idea" or the use of "yes, and...")
4. **Stay focused on the topic at hand**
5. **Have one conversation at a time**
6. **Be visual** (Draw and/or upload to show ideas, whenever possible.)
7. **Go for quantity**

C **Interested in learning more?**
Check out the Meta Think Kit website for additional tools and resources to help your team collaborate, innovate and move ideas forward with confidence.

Open the website →

1 Choose your best "How Might We" Questions

Share the top 5 brainstorm questions that you created and let the group determine where to begin by selecting one question to move forward with based on what seems to be the most promising for idea generation in the areas you are trying to impact.

10 minutes

QUESTION
what do we incorporate to make it a smart signboard

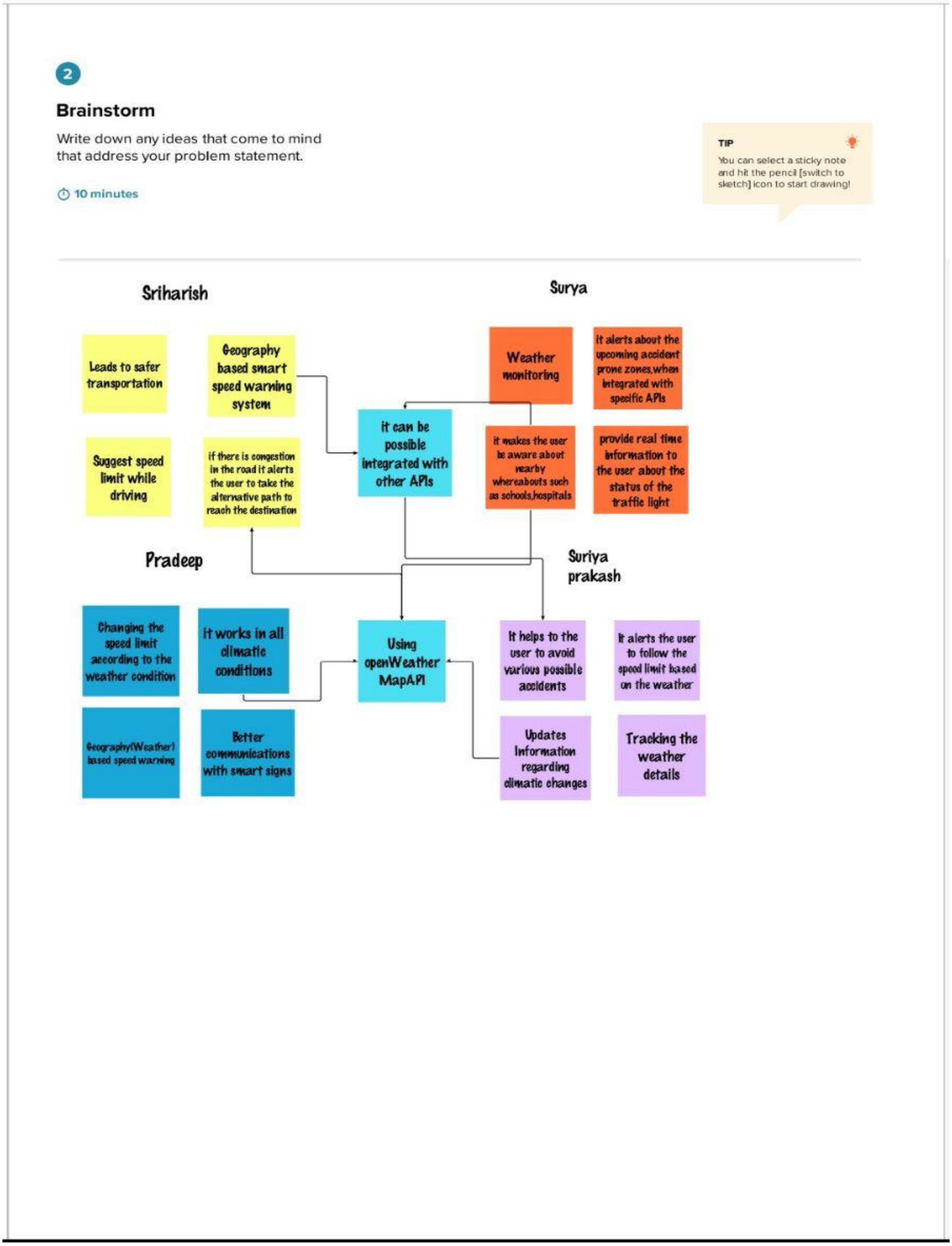
QUESTION
how helpful is the smart signboards

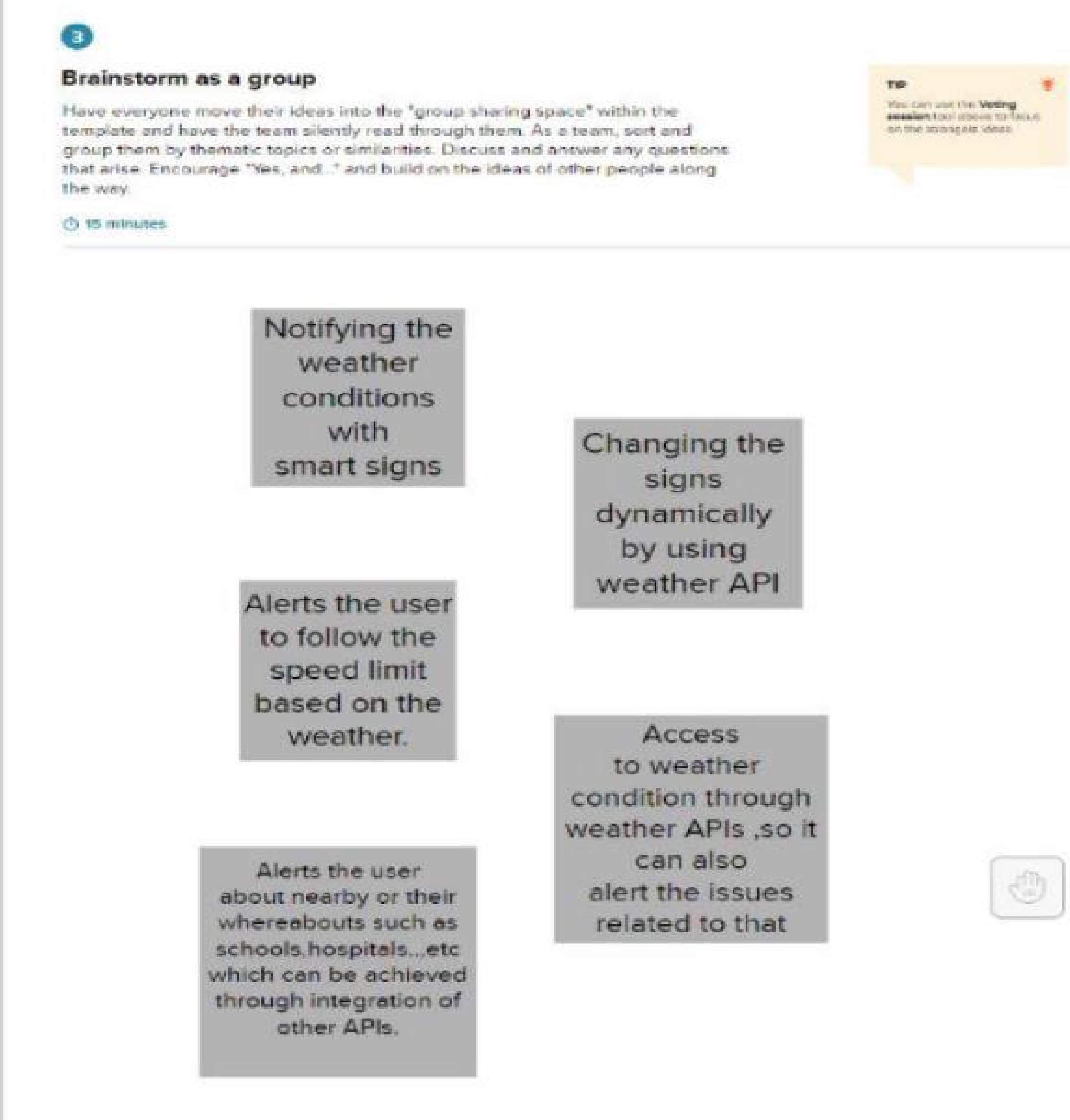
QUESTION
In what minimum ways it can be achieved

QUESTION
what can be done

Step 2:Idea

Prioritization:





3.3 PROPOSED SOLUTION

S.No.	Parameter	Description
1.	Problem Statement (Problem to be solved)	To avoid road accidents caused by over speeding of vehicles at the time of bad weather conditions like heavy rain,high wind

2.	Idea / Solution description	The project approach to digitizing the already existing static signboards to smart signboards using a interface where people are able to see about weather the indications and recommend the speed limit.This information can be accessed from open weather map and we can display the updateson the user interface on a timely basis. The smart display gets the speed limitations from a web app using weather API.
3.	Novelty / Uniqueness	Sign boards are converted to digital display where APIs and online services are integrated in new and interesting ways. Open Weather Map is an online service that provides global weather data, forecasts and historical weather data for any geographical location.
4.	Social Impact / Customer Satisfaction	It's suggests the speed limits for the user/driver based on the weather and this will be helpful to reduce accidents meet by bad weather .
5.	Business Model (Revenue Model)	It will be implemented in low cost(for specified limited area not for large scale).In low cost we can reduce the accident

		possibility and save many of them life.
6.	Scalability of the Solution	This project is highly useful and later on be further updated and additional features will be added well .

3.4 PROBLEM SOLUTION FIT

Define CS, fit into CC

1. CUSTOMER SEGMENT(S) CS

A traveller/driver who wants to travel and reache their destination safely with any trouble or accident possibilities

6. CUSTOMER CONSTRAINTS CC

Proper network connection, Data Accuracy, Availability of proper device

5. AVAILABLE SOLU

By replacing the static signboards, smart connected signboards are used. Based

Focus on J&P, tap into BE, understand RC

2. JOBS-TO-BE-DONE / PROBLEMS J&P

To provide better safety for travellers by providing information about traffic and speed limit by analysing the weather conditions.

9. PROBLEM ROOT CAUSE RC

Lack of proper awareness about road safety and no proper knowledge about traffic rules.

7. BEHAV

User n technolo informati

Identify strong TR & EM

3. TRIGGERS

Fear of accidents . By what they see during the travel and hearing news about accidents who travel without proper precautions.

4. EMOTIONS: BEFORE / AFTER

Wants to travel on the road -> About to make an accident -> feeling fearful -> follows traffic rules and take necessary precautions.

10. YOUR SOLUTION

These smart connected sign boards get the speed limitations from a web app using weather API and update automatically.
Based on the weather changes the speed may increase or decrease.
Based on the traffic and fatal situations the diversion signs are displayed.
Guide(Schools), Warning and Service(Hospitals, Restaurant) signs are also displayed accordingly. Different modes of operations can be selected with the help of buttons.

8. CHANNELS of BEHAVIOUR

CH

ONLINE: Calling customer care, Mailing

OFFLINE: Going to service center

Identify strong TR & EM

CHAPTER 4
REQUIREMENT ANALYSIS

Functional Requirements:

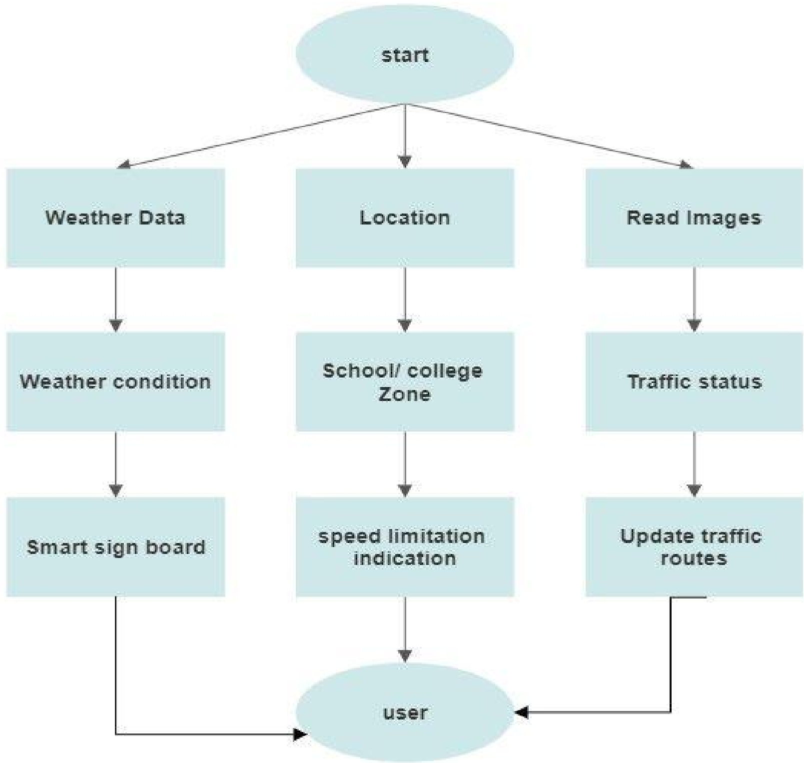
FR No.	**Functional Requirement**	**Sub Requirement (Story / Sub-Task)**
FR-1	**User Visibility**	Sign Boards should be made with LED's which are bright colored and are capable of attracting the drivers attention
FR-2	**User Need**	The smart sign boards should be placed in location where the accident happens frequently.
FR-3	**User Understanding**	For better understanding of the driver, the signs should be big, clear .
FR-4	**User Convenience**	The display should be big enoughthat should be easy to understand.

Non-functional Requirements:

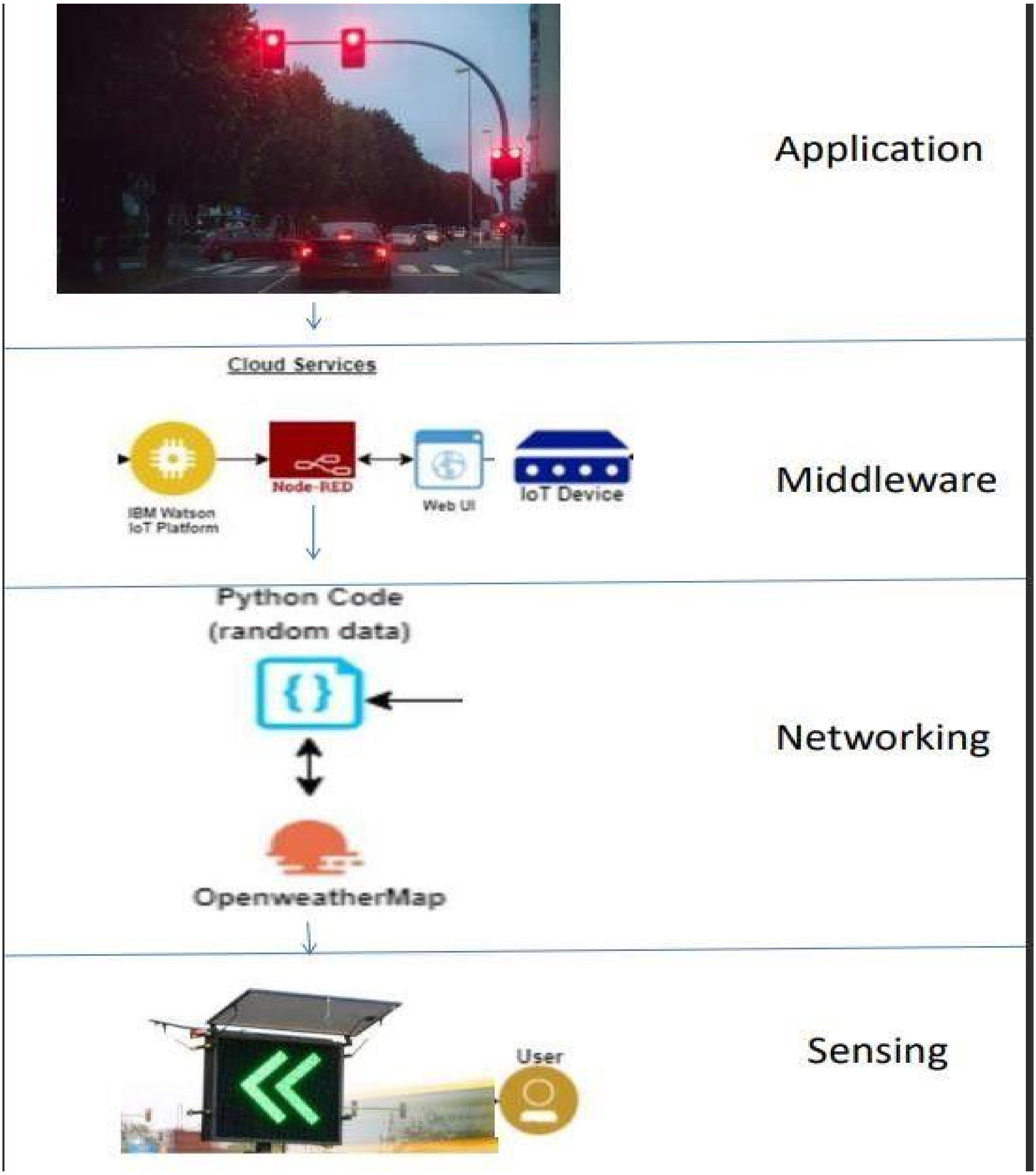
FR No.	Non-Functional Requirement	Description
NFR- 1	**Usability**	It should be able to Upgrade and Update .
NFR-2	**Security**	It should have good security and user information should be secured
NFR-3	**Reliability**	It should be able to display to correct information
NFR-4	**Performance**	It should be able to automatically update itself when certain weather Or traffic problem occurs.
NFR-5	**Availability**	It should be available 24/7 for the user
NFR-6	**Scalability**	It should be able to easily change and upgrade based on the needs

CHAPTER 5
PROJECT DESIGN

5.1 DATA FLOW DIAGRAM:



5.2.Technology Architecture:



GUIDELINES:

To replace the static signboards, smart connected sign boards areused. These smart connected sign boards get the speed limitations froma web app using weather API and update automatically. Based on the weather changes the speed may increase ordecrease. Based on the traffic and fatal situations the diversion signs aredisplayed. Guide(Schools), Warning and Service(Hospitals, Restaurant)signs are also displayed accordingly. Different modes of operations can be selected with the help of buttons.

5.3 User Stories:

User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
Customer (Mobile user)	Registration	USN-1	I can get my speed limitation using weather application.	I can receive speed limitations	High	Sprint-1
		USN-2	As a user, I can register for the applicationby entering my email, password, andconfirming my password. As a user,	I can access my account /dashboard	Medium	Sprint-2
		USN-3	As a user, I can increase or decrease myspeed according to the weather change	I can increase or decrease my speed	High	Sprint-1
		USN-4	As a user, I can I get my traffic diversion signs depending on the traffic and the fatalsituations.	I can access my traffic status ahead in my travel	Medium	Sprint-1
	Login	USN-5	As a user, I can log into the open weathermap by entering email & password	I can access the application through my Gmail login	High	Sprint-2
Customer (Webuser)	Data generation	USN-6	As a user I use open weather application to access the data regarding the weather changes.	I can access the data regarding the weather through the	High	Sprint-1

				application		
Administrator (Officials)	Problem solving/ Fault clearance	USN-7	As an official who is in charge for the proper functioning of the sign boards have to maintain itthrough periodic monitoring.	Officials can monitor thesign boards for proper functioning.	Medium	Sprint-2

CHAPTER 6

6.1 SPRINT PLANNING AND ESTIMATION :

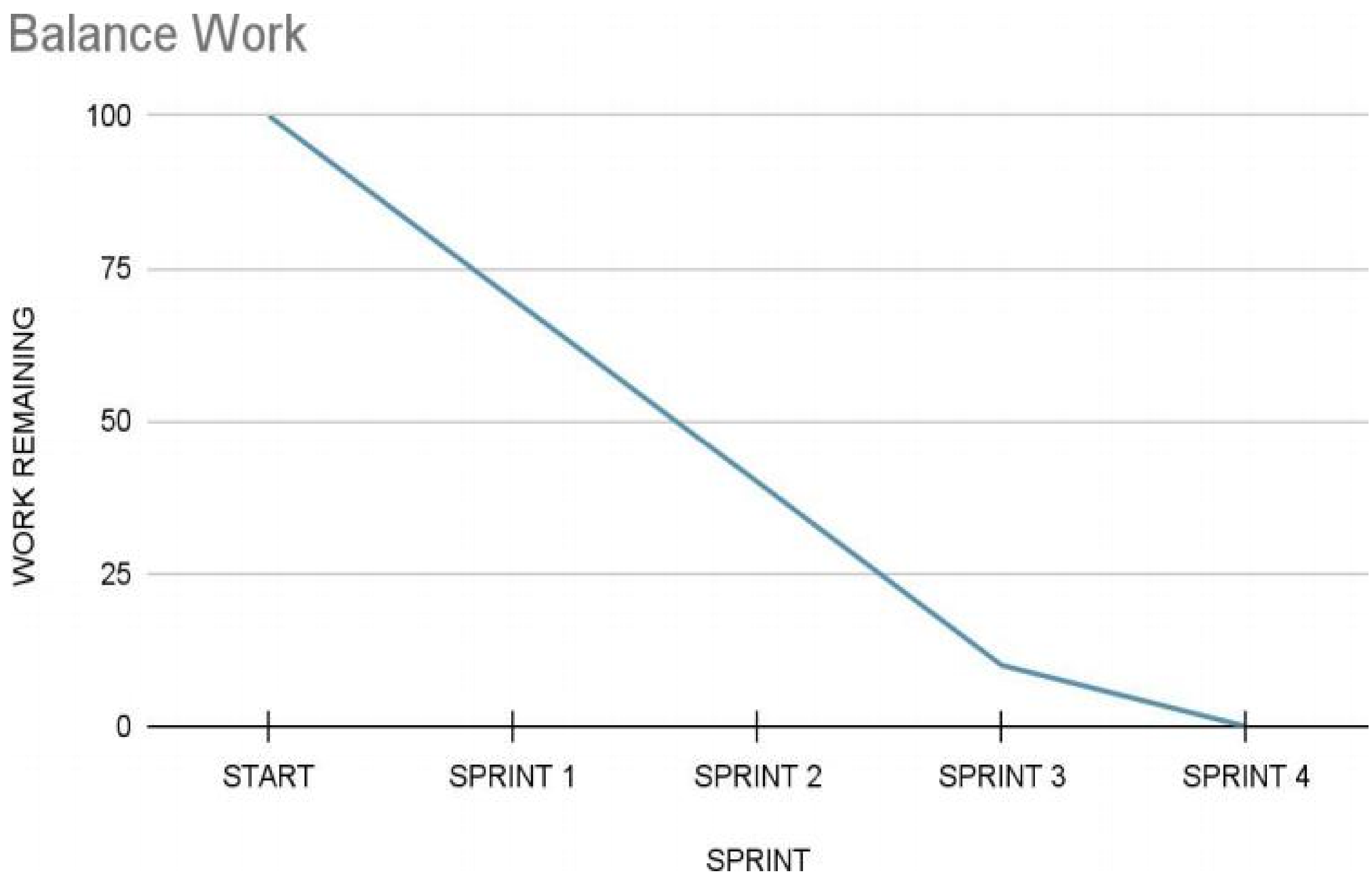
Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	20	6 Days	24 Oct 2022	29 Oct 2022	20	29 Oct 2022
Sprint-2	20	6 Days	31 Oct 2022	05 Nov 2022	20	05 Nov 2022
Sprint-3	20	6 Days	07 Nov 2022	12 Nov 2022	20	12 Nov 2022
Sprint-4	20	6 Days	14 Nov 2022	19 Nov 2022	20	19 Nov 2022

6.2 SPRINT DELIVERY SCHEDULE:

Sprint	Functional Requirement (Epic)	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Resources Initialization	Create and initialize accounts using Open Weather API	1	Low	Sriharish PK, Surya S,Suriya prakash, Pradeep G
Sprint-1	Local Server/Software Run	Write a Python program that outputs results given the inputs like weather and location	1	Medium	Sriharish PK, Surya S,Suriya prakash, Pradeep G
Sprint-2	Push the software/server to cloud	Push the code from Sprint 1 to cloud so it can be accessed from anywhere	2	Medium	Sriharish PK, Surya S,Suriya prakash, Pradeep G
Sprint-3	Hardware Initialization	Integrate the hardware to be able to access the cloud functions and provide inputs to the same.	2	Medium	Sriharish PK, Surya S,Suriya prakash, Pradeep

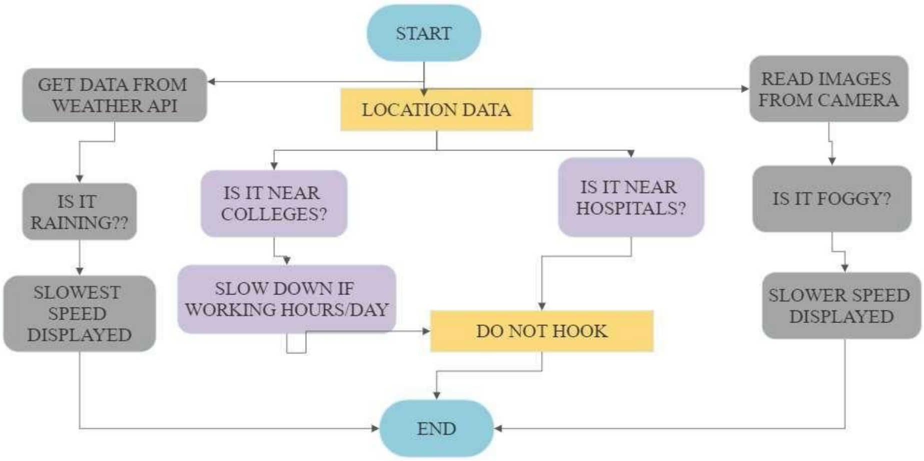
					G
Sprint-4	Login	Optimize all the defects and provide better user experience.	2	High	Sriharish PK, Surya S,Suriya prakash, Pradeep G

Burndown Chart:



CHAPTER-7
CODING AND SOLUTIONING

CODE FLOW:



SPRINT 1:

Weather.py

This file is a utility function that fetches the weather from Open WeatherAPI. Itreturns only certain required parameters of the API response.

```
# Python code
import requests as reqs
def get(myLocation,APIKEY):
apiURL =
f"https://api.openweathermap.org/data/2.5/weather?q={myLocatio
n}&appid={APIKEY}"
responseJSON = (reqs.get(apiURL)).json()
returnObject = {
"temperature" : responseJSON['main']['temp'] - 273.15,
"weather" : [responseJSON['weather'][_]['main'].lower() for _
in range(len(responseJSON['weather']))],
"visibility" : responseJSON['visibility']/100, # visibility in
```

percentage where 10km is 100% and 0km is 0%

}

if("rain" in responseJSON):

returnObject["rain"] = [responseJSON["rain"][key] for key in responseJSON["rain"]]

return(returnObject)

Brain.py

This file is a utility function that returns only essential information to be displayed at the hardware side and abstracts all the unnecessary details. This is where the code flow logic is implemented.

```
import weather
from datetime import datetime as dt
# IMPORT SECTION ENDS
# ----------------------------------------------
# UTILITY LOGIC SECTION STARTS
def processConditions(myLocation,APIKEY,localityInfo):
weatherData = weather.get(myLocation,APIKEY)
finalSpeed = localityInfo["usualSpeedLimit"] if "rain" not in
weatherData else localityInfo["usualSpeedLimit"]/2
finalSpeed = finalSpeed if weatherData["visibility"]>35 else
finalSpeed/2
if(localityInfo["hospitalsNearby"]):
# hospital zone
doNotHonk = True
else:
if(localityInfo["schools"]["schoolZone"]==False):
# neither school nor hospital zone
doNotHonk = False
else:
```

```
# school zone
now = [dt.now().hour,dt.now().minute]
activeTime = [list(map(int,_.split(":"))) for _ in
localityInfo["schools"]["activeTime"]]
doNotHonk = activeTime[0][0]<=now[0]<=activeTime[1][0] and
activeTime[0][1]<=now[1]<=activeTime[1][1]
return({
"speed" : finalSpeed,
"doNotHonk" : doNotHonk
})
# UTILITY LOGIC SECTION ENDS
```

Main.py

The code that runs in a forever loop in the microcontroller. This calls all the util functions from other python files and based on the return value transduces changes in the output hardware display.

```
# Python code
# IMPORT SECTION STARTS
import Brain
# IMPORT SECTION ENDS
# -----------------------------------------------
# USER INPUT SECTION STARTS
myLocation = "Coimbatore,IN"
APIKEY = "9cd610e5fd400c74212074c7ace0d62c"
localityInfo = {
"schools" : {
"schoolZone" : True,
"activeTime" : ["9:00","16:00"] # schools active from 7 AM till 5:30
PM
},
```

```
"hospitalsNearby" : False,
"usualSpeedLimit" : 45 # in km/hr
}
# USER INPUT SECTION ENDS
# ---------------------------------------------
# MICRO-CONTROLLER CODE STARTS
print(Brain.processConditions(myLocation,APIKEY,localityInfo))
''' MICRO CONTROLLER CODE WILL BE ADDED IN SPRINT 2 AS
PER OUR PLANNED SPRINT SCHEDULE
''' # MICRO-CONTROLLER CODE ENDS
```

SPRINT 2:

1.Login

```
<?php
$pass=$_POST['password'];
$username=$_POST['username'];
$dbc = mysqli_connect('mysql.hostinger.in', 'u684030433_root', 'fastrack',
'u684030433_blood')
or
die("error connecting database");
$query = "SELECT username,name FROM user_info WHERE
((username = '$username' OR
email='$username' )AND password ='$pass')";
$data = mysqli_query($dbc, $query);
if (mysqli_num_rows($data) == 1) {
$name=array();
while($row=mysqli_fetch_array($data))
{
$name[]=$row['name'];
```

```
}
print json_encode($name);
}
else
{
echo "Wrong";
}
```

Date

03 Nov 2022

Team ID

PNT2022TMID12612

Project Name

Project – Signs with Smart Connectivity for Better Road Safety

```
mysqli_close($dbc);
?>
```

2. SAFETY introduction

```
<?php
$lat1="";
$lat2="";
$lng1="";
$lng2="";
function distance($lat1, $lng1, $lat2, $lng2) {
$theta = $lng1 - $lng2;
$dist = sin(deg2rad($lat1)) * sin(deg2rad($lat2)) + cos(deg2rad($lat1)) *
cos(deg2rad($lat2))
*
cos(deg2rad($theta));
$dist = acos($dist);
$dist = rad2deg($dist);
```

```
$miles = $dist * 60 * 1.1515;
return ($miles * 1.609344*1000);
}
if($_GET['username'])
{
$username=$_GET['username'];
$lat1=(double)$_GET['lat'];
$lng1=(double)$_GET['lng'];
$dbc = mysqli_connect('localhost', 'root', '', 'roadsafety');
// Retrieve the score data from MySQL
$query = "UPDATE user_info SET lat='$lat1',lng='$lng1' WHERE (username =
'$username')";
mysqli_query($dbc, $query)
or die('Error querying database.');
$query = "SELECT lat,lng FROM user_info WHERE username='$username'";
$data = mysqli_query($dbc, $query);
while ($row = mysqli_fetch_array($data)) {
$lat1= $row['lat'];
$lng1= $row['lng'];
}
$query1 = "SELECT * FROM details";
$result = mysqli_query($dbc, $query1);
$i=0;
$data1=array();
while ($row = mysqli_fetch_array($result, MYSQL_ASSOC)) {
$lat2= $row['lat'];
$lng2= $row['lng'];$dist=distance($lat1, $lng1, $lat2, $lng2);
```

```
if( $dist<=990)
{
$data1[] = $row;
$data1[$i]['metres']=$dist;
$i++;
}}
echo json_encode($data1);
mysqli_close($dbc);
}
?>
```

3. Signin codes

```
<?php
$email= $_POST['email'];
$pass=$_POST['password'];
$name=$_POST['name'];
$username=$_POST['username'];
$pno=(double)$_POST['pno'];
$bloodgroup=$_POST['bloodgroup'];
$dbc = mysqli_connect('mysql.hostinger.in', 'u684030433_root', 'fastrack',
'u684030433_blood')
or
die("error connecting database");
$query = "SELECT * FROM user_info WHERE (username =
'$username' OR
email='$email'
OR pno='$pno')";
$data = mysqli_query($dbc, $query);
if (mysqli_num_rows($data) == 0) {
```

```
$query = "INSERT INTO user_info
(username,password,email,pno,bloodgroup,name)
VALUES ('$username', '$pass', '$email','$pno','$bloodgroup','$name')";
mysqli_query($dbc, $query)
or die('Error querying database.');
echo 'User Signed Up successfully!!.';
}
else
{
echo "Account already exists for this credentials...";
}
mysqli_close($dbc);
?>
```

4 build.gradle

```
def localProperties = new Properties()
def localPropertiesFile = rootProject.file('local.properties')
if (localPropertiesFile.exists()) {
localPropertiesFile.withReader('UTF-8') { reader ->
localProperties.load(reader)
}}
def flutterRoot = localProperties.getProperty('flutter.sdk')
if (flutterRoot == null) {
throw new GradleException("Flutter SDK not found. Define location
with flutter.sdk in the
local.properties file.")
}
def flutterVersionCode =
localProperties.getProperty('flutter.versionCode')
if (flutterVersionCode == null) {
```

```
flutterVersionCode = '1'
}
def flutterVersionName =
localProperties.getProperty('flutter.versionName')
if (flutterVersionName == null) {
flutterVersionName = '1.0'
}
apply plugin: 'com.android.application'
apply plugin: 'com.google.gms.google-services'
apply plugin: 'kotlin-android'
apply from: "$flutterRoot/packages/flutter_tools/gradle/flutter.gradle"
android {
compileSdkVersion 28
sourceSets {
main.java.srcDirs += 'src/main/kotlin'
}
lintOptions {
disable 'InvalidPackage'
}
defaultConfig { // TODO: Specify your own unique Application ID
(https://developer.android.com/studio/build/application-id.html).
applicationId "com.example.roads"
minSdkVersion 18
targetSdkVersion 28
multiDexEnabled true
versionCode flutterVersionCode.toInteger()
versionName flutterVersionName
buildConfigField 'String', 'WONDERPUSH_CLIENT_ID',
'"1dfce26a84bd50a2b2117ae3a65df6f3d08821cb"'
```

```
buildConfigField 'String', 'WONDERPUSH_CLIENT_SECRET',
'"d490ca2368fb19536c8e5afc370d18e9002a5034bf14e13e4b4a9228dd39
c5a0"'
buildConfigField 'String', 'WONDERPUSH_SENDER_ID',
'"1098204096327"'
}
buildTypes {
release {
// TODO: Add your own signing config for the release build.
// Signing with the debug keys for now, so `flutter run --release` works.
signingConfig signingConfigs.debug
}}}
flutter {
source '../..'
}
dependencies {
implementation "org.jetbrains.kotlin:kotlin-stdlib-jdk7:$kotlin_version"
implementation platform('com.google.firebase:firebase-bom:26.1.0')
implementation 'com.google.firebase:firebase-analytics'
def multidex_version = "2.0.1"
implementation 'androidx.multidex:multidex:$multidex_version'
}
```

SPRINT 3:

Application Packages :

```
package com.example.myhp.accidentprevention;
import android.app.Application;
import android.test.ApplicationTestCase;
/*
```

<ahref="http://d.android.com/tools/testing/testing_android.html">Testing Fund>
*/
public class ApplicationTest extends ApplicationTestCase<Application>
{
public ApplicationTest()
{
super(Application.class);
}}
MAIN :
\# Manipulating data {#data}
This section is an introduction to manipulating datasets using the `dplyr` package.
As outlined in the previous section, `dplyr` and `ggplot2` are part of the `tidyverse`,
which aims
to provide a user-friendly framework for data science [@grolemund_data_2016].
Date
03 Nov 2022
Team ID
PNT2022TMID12612
Project Name
Project – Signs with Smart Connectivity for Better Road SafetyExperience of teaching R
over the past few years suggests that many people find it easier to get
going with data driven research if they learn the 'tidy' workflow presented in this
section.

However, if you do not like this style of R code or you are simply curious, we
encourage you to
try alternative approaches for achieving the similar results using base R [@rcoreteam_language_2020]^[
Run the command `help.start()` to see a resources introducing base R, and [Chapter 6
on lists and
data frames](https://cran.r-project.org/doc/manuals/r-release/R-intro.html#Lists-and data-frames)
in [An Introduction to R](https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf)
in
particular for an introduction to data manipulation with base R.
]
, the `data.table` R package [@R-data.table] or other languages such as [Python](https://www.python.org/) or [Julia](https://julialang.org/).
If you just want to get going with processing data, the `tidyverse` is a solid and
popular starting
point.
<!-- Todo: add new part here? -->
Before diving into the `tidyverse`, it is worth re-capping where we have got to so far
as we have
covered a lot of ground.
Section \@ref(basics) introduced R's basic syntax; Section \@ref(rstudio) showed
how to use the

Source Editor and other features of RStudio to support data science; and Section
\@ref(pkgs)
introduced the concept and practicalities of R packages, with reference to `stats19`,
`ggplot2` and
`dplyr`.
In this section, we will start with a blank slate.
In Section \@ref(basics) we learned that in R having a 'clear desk' means an *empty
global
environment*.
This can be achieved by running the following command, which removes the `list()`
of all
objects returned by the function `ls()`:
{r}
rm(list = ls())
tibbles
Although the data processing techniques in R are capable of handling large datasets,
such as the
`crashes_2019` object that we created in the previous section, representing 100k+
casualties, it
makes sense to start small.
Let's start by re-creating the `crashes` dataset from Section \@ref(basics), but this
time using the

`tidyverse` `tibble()` function. This is the `tidyverse` equivalent of base R's
`data.frame`.
`tibble` objects can be created, after loading the `tidyverse`, as follows:

```
{r, message=FALSE}
library(tidyverse)
crashes = tibble( casualty_type = c("pedestrian", "cyclist", "cat"),
casualty_age = seq(from = 20, to = 60, by = 20),
vehicle_type = c("car", "bus", "tank"),
dark = c(TRUE, FALSE, TRUE)
)
```

In the previous code chunk, we passed four vector objects as named arguments to
the `tibble`
function, resulting in columns such as `casualty_type`.
A `tibble` is just a fancy way of representing `data.frame` objects, preferred by
`tidyverse` users
and optimised for data science.
It has a few sensible defaults and advantages compared with the `data.frame`, one of
which can
be seen by printing a `tibble`:

```
{r}
class(crashes)
crashes
```

Note the `<chr>`, `<dbl>` or `<lgl>` text below each column, providing a quick

indication of the
class of each variable - this is not provided when using `data.frame`.
filter() and select() rows and columns
In the previous section, we briefly introduced the package `dplyr`, which provides an
alternative
to base R for manipulating objects. `dplyr` provides different, and some would argue
simpler,
approaches for subsetting rows and columns than base R.
`dplyr` operations for subsetting rows (with the function `filter()`) and columns (with
the
function `select()`) are demonstrated below. Here we can also see the use of the pipe
operator
`%>%` to take the dataset and apply the function to that dataset.
{r}
crashes %>% filter(casualty_age > 50) # filters rows
crashes %>% select(casualty_type) # select just one column

It should be clear what happened: `filter()` returns only rows that match the criteria in
the
function call, only observations with a `casualty_age` greater than 50 in this case.
Likewise, `select()` returns data objects that include only columns named inside the
function call,

`casualty_type` in this case.

To gain a greater understanding of the functions, type and run the following
commands, which
also illustrate how the `%>%` can be used more than once to manipulate data (more
on this
soon):

{r}crashes_darkness = crashes %>% filter(dark)
crashes_a = crashes %>% select(contains("a"))
crashes_darkness_a = crashes %>%
filter(dark) %>%
select(contains("a"))

Can you guess what the dimensions of the resulting objects will be?
Write down your guesses for the number of rows and number of columns that the new
objects,
`crashes_darkness` to `crashes_darkness_a`, have before running the following
commands to
find out. This also demonstrates the handy function `dim()`, short for dimension
(results not
shown):^[
Note that the number of rows is reported before the number of columns.
This is a feature of R: rows are also specificiefied first when subsetting using the
square brackets
in commands such as `crashes[1, 2:3]`.

]
{r, eval=FALSE}
dim(crashes)
dim(crashes_darkness)
?contains # get help on contains() to help guess the output of the next line
dim(crashes_a)
dim(crashes_darkness_a)

Look at the help pages associated with `filter()`, `select()` and the related function
`slice()` as
follows and try running the examples that you will find at the bottom of the help
pages for each
to gain a greater understanding (note you can use the `package::function` notation to
get help on
functions also):

{r, eval=FALSE}
?dplyr::filter
?dplyr::select
?dplyr::slice

Ordering and selecting the 'top n'

Other useful pipe-friendly functions are `arrange()` and `top_n()`. `arrange()` can be
used to sort
data. Within. the `arrage()` function, optional arguments can be used to define the
order in which

it is sorted. `top_n()`simply selects the top 'n' number of rows in your data frame.

We can use these functions to arrange datasets and take the top most ‘n’ values, as
follows:

{r}

crashes %>% arrange(vehicle_type)

crashes %>%

top_n(n = 1, wt = casualty_age)

<!-- ## Long and wide data -->

Summarise

A powerful two-function combination is `group_by()` and `summarise()`. Used together, they can provide grouped summaries of datasets.

In the example below, we find the mean age of casualties in dark and light conditions.

{r}

crashes %>%

group_by(dark) %>%

summarise(mean_age = mean(casualty_age))

The example above shows a powerful feature of these pipelines. Many operations can
be
'chained' together, whilst keeping readability with subsequent commands stacked
below earlier
operations. The combination of `group_by()` and `summarise()` can be very useful in
preparing
data for visualisation with a `ggplot2` function.

Another useful feature of the `tidyverse` from a user perspective is the autocompletion
of column
names mid pipe.
If you have not noticed this already, you can test it by typing the following, putting
your cursor
just before the `)` and pressing `Tab`:
{r, eval=FALSE}
crashes %>% select(ca) # press Tab when your cursor is just after the a
You should see `casualty_age` and `casualty_type` pop up as options that can be
selected by
pressing `Up` and `Down`.
This may not seem like much, but when analysing large datasets with dozens of
variables, it can
be a godsend.
Rather than providing a comprehensive introduction to the `tidyverse` suite of
packages, this
section should have offered enough to get started with using it for road safety data
analysis.
For further information, check out up-to-date online courses from respected
organisations like
[Data Carpentry](https://datacarpentry.org/R-ecology-lesson/index.html) and the free

online
[books](https://bookdown.org/) such as [R for Data Science]((https://r4ds.had.co.nz/))
[@grolemund_data_2016].
Tidyverse exercises1. Use `dplyr` to filter rows in which `casualty_age` is less than
18, and then 28.
2. Use the `arrange` function to sort the `crashes` object in descending order of age
(*Hint:*
see the `?arrange` help page).
3. Read the help page of `dplyr::mutate()`. What does the function do?
4. Use the mutate function to create a new variable, `birth_year`, in the `crashes`
data.frame
which is defined as the current year minus their age.
5. *Bonus:* Use the ` %>% ` operator to filter the output from the previous
exercise so that
only observations with `birth_year` after 1969 are returned.

```
{r dplyr, eval=FALSE, echo=FALSE}
# answers
crashes %>%
arrange(desc(casualty_age))
crashes %>% filter(casualty_age > 21)
crashes %>%
mutate(birth_year = 2019 - casualty_age) %>%
filter(birth_year > 1969)
```

Slides :

```
---
title: " Road Safety (and transport) Research with R"
# subtitle: '`r emojifont::emoji("bike")`<br/>For England and Wales'
subtitle: '`r emojifont::emoji("rocket")`<br/>RAC Foundation, Data
Driven'
author: "Robin Lovelace"
date: '2020'
output:
xaringan::moon_reader:
# css: ["default", "its.css"]
# chakra: libs/remark-latest.min.js
lib_dir: libs
nature:
highlightStyle: github
highlightLines: true
# bibliography:
# - ../vignettes/ref.bib
# - ../vignettes/ref_training.bib
---
{r setup, include=FALSE, eval=FALSE}
# get citations
refs = RefManageR::ReadZotero(group = "418217", .params =
list(collection =
"JFR868KJ",
limit = 100))
refs_df = as.data.frame(refs)
# View(refs_df)# citr::insert_citation(bib_file =
"vignettes/refs_training.bib")
RefManageR::WriteBib(refs, "refs.bib")
```

```
# citr::tidy_bib_file(rmd_file = "vignettes/pct_training.Rmd",
messy_bibliography =
"vignettes/refs_training.bib")
options(htmltools.dir.version = FALSE)
knitr::opts_chunk$set(message = FALSE)
library(RefManageR)
BibOptions(check.entries = FALSE,
bib.style = "authoryear",
cite.style = 'alphabetic',
style = "markdown",
first.inits = FALSE,
hyperlink = FALSE,
dashed = FALSE)
my_bib = refs

{r, eval=FALSE, echo=FALSE, engine='bash'}
# publish results online
cp -Rv code/rrsrr-slides* ~/saferactive/site/static/slides/
cp -Rv code/libs ~/saferactive/site/static/slides/
cd ~/saferactive/site
git add -A
git status
git commit -am 'Update slides'
git push
cd -
 # Slide/links
https://itsleeds.github.io/rrsrr/
https://bookdown.org/
```

https://www.pct.bike/ --- background-image: url(https://media.giphy.com/media/YlQQYUIEAZ76o/giphy.gif)
Coding
Ideal:{r, eval=FALSE}
od_test$perc_cycle = round(od_test$bicycle / od_test$all) * 100
l = od_to_sf(od_test, od_data_centroids)
r = stplanr::route(l = l, route_fun = journey)
rnet = overline(r, "bicycle")

--
`![](https://media.giphy.com/media/3oKIPnAiaMCws8nOsE/giphy.gif)`
Reality
--- ## Transport software - which do you use?
{r, echo=FALSE, message=FALSE, warning=FALSE}
u = "https://github.com/ITSLeeds/TDS/raw/master/transport-software.csv"
tms = readr::read_csv(u)[1:5]
tms = dplyr::arrange(tms, dplyr::desc(Citations))
knitr::kable(tms, booktabs = TRUE, caption = "Sample of transport modelling
software in use by
practitioners. Note: citation counts based on searches for company/developer name,
the product
name and 'transport'. Data source: Google Scholar searches, October 2018.", format =
"html")

--- ## Data science and the tidyverse

- Inspired by Introduction to data science with R (available free
[online](https://r4ds.had.co.nz/))
{r tds-cover, echo=FALSE, out.width="30%"}
knitr::include_graphics("https://d33wubrfki0l68.cloudfront.net/b88ef926a
004b0fce72
b2526b0b5
c4413666a4cb/24a30/cover.png")

--- ## A geographic perspective
- See https://github.com/ITSLeeds/TDS/blob/master/catalogue.md
- Paper on the *stplanr* paper for transport planning (available
[online](https://cran.r-project.org/web/packages/stplanr/vignettes/stplanr-paper.html))
- Introductory and advanced content on geographic data in R, especially the [transport
chapter](https://geocompr.robinlovelace.net/transport.html) (available free
[online](https://geocompr.robinlovelace.net/))
- Paper on analysing OSM data in Python with OSMnx (available
[online](https://arxiv.org/pdf/1611.01890))
--- # Getting support --With open source software, the world is your support network!
--
- Recent example: https://stackoverflow.com/questions/57235601/ --
- [gis.stackexchange.com](https://gis.stackexchange.com/questions) has 21,314
questions
- [r-sig-geo](https://r-sig-geo.2731867.n2.nabble.com/) has 1000s of posts

- RStudio's Discourse community has 65,000+ posts already!

--

- No transport equivalent (e.g. earthscience.stackexchange.com is in beta)
- Potential for a Discourse forum or similar: transport is not (just) GIS

--- Textcenter_analysis:

--- pagetitle: "Analysis based on test centres"

author: "Amol Nanaware"

date: "06/12/2019"

always_allow_html: true

output:

html_document: default

---{r setup, include=FALSE}

knitr::opts_chunk$set(echo = TRUE)

{r, warning=FALSE, echo=FALSE, include=FALSE}

packages <- c("plotly", "tidyverse")

newPackages <- packages[!(packages %in% installed.packages()[,"Package"])]

if(length(newPackages)) install.packages(newPackages)

library(tidyverse)

library(plotly)

{r,echo=FALSE}

load("passfail.RData")

passfail <- passfail %>%

mutate(totalFails = Fail1 + ifelse(is.na(Fail2), 0, Fail2), Totalpass = Pass1 +

ifelse(is.na(Pass2),

0, Pass2))

```
{r,echo=FALSE}
passfailGroup <- summarise(group_by(passfail, Centre), Pass1 =
sum(Pass1), Fail1 =
sum(Fail1), Total1 = sum(Total1), Pass2 = sum(Pass2, na.rm = T), Fail2
= sum(Fail2,
na.rm =
T), Total2 = sum(Total2, na.rm = T), Totalpass = sum(Totalpass),
totalFails =
sum(totalFails))
passfailGroup <- mutate(passfailGroup, Pass1prop = Pass1/Total1,
Pass2prop =
Pass2/Total2,
totalPassProp = (Totalpass / (Total1 + Total2)), totalFailsProp =
(totalFails / (Total1 +
Total2)))
```

```
{r,echo=FALSE}
passfailGroup$totalPassProp = round((passfailGroup$totalPassProp *
100), digits = 2)
passfailGroup$totalFailsProp = round((passfailGroup$totalFailsProp *
100), digits =
2)
passFailGroup1 <- passfailGroup[c(1, 8)]
passFailGroup1$Test <- "Pass"
names(passFailGroup1) <- c("Centre", "Count", "Test")
passFailGroup2 <- passfailGroup[c(1, 9)]
passFailGroup2$Test <- "Fail"
names(passFailGroup2) <- c("Centre", "Count", "Test")
```

passFailcount <- rbind(passFailGroup1, passFailGroup2)
Analysis based on test centres

In this section we will analyse data from 2013 till 2018 about each test centre.
As shown in
the map, the test centres
are spread
across the Ireland and the number of centres is more in highly populated areas such as
dublin,
cork etc.
The bar chart shows the total number of tests that each centre performed and the total
pass and
fail counts as well as percentages. So, based on the test counts, the top 3 test centre
are,
Fonthill(770685), Deansgrade(767484), and Northpoint 2(729661). The botton
3 centres
which performed less tests are, Donegal Town(16315), Cahirciveen(28806) and
Clifden(38683).
{r,echo=FALSE, fig.width=9,fig.height=4}
p <- plot_ly(passfailGroup, x = ~passfailGroup$Centre, y = ~passfailGroup$Totalpass,
type =

'bar', name = 'Pass', text = paste("Total tests = ", (passfailGroup$Totalpass+passfailGroup$totalFails), "
Passed =", passfailGroup$totalPassProp,"%", "
Failed =", passfailGroup$totalFailsProp,"%"),
opacity =
0.5, marker = list(color = '#3AC3E3', line = list(color = '#0D6EB0', width = 1))) %>%
add_trace(y = ~passfailGroup$totalFails, name = 'Fails', opacity = 0.5, marker = list(color = '#0E84FF', line = list(color = '#0D6EB0', width = 1))) %>%
layout(yaxis = list(title = 'Count'), xaxis = list(title = 'Test Centres'), barmode = 'stack')
p
Total test passed for each test centre
The following scatter plot show the total test pass count for each test centre from the
year 2013
till year 2018. The questions that can be answered by this graph are,
1. which are the top 3 and last 3 centres based on total pass count?
(Deansgrade, Northpoint 2, Fonthill and
Cahirciveen, Clifden, derrybeg resp.)
2. Which year has the highest and lowest total pass count?
2015 and 2014 respectively
But, in this graph we are not considering the total tests performed by the test centres
which
shows the actual performance of the tests. For this we will plot another graph.

{r,echo=FALSE, fig.width=9,fig.height=4}
#scatter plot for centre total pass per year
ggplot(data = passfail, aes(x = fct_reorder(Centre, -Totalpass), y = Totalpass, color =
Year, size
= Totalpass)) + geom_point(alpha = 0.5) +
theme(axis.text.x = element_text(size=9, angle=-90, hjust = 0, vjust = 0.5),
axis.ticks.x =
element_blank(), panel.background = element_rect(fill = "white", colour =
"lightblue"),
panel.grid.minor = element_line(size = 0.5, linetype = 'solid', colour = "lightblue")) +
labs(x = "Test Centres", y = "Totol pass count")
Test performance for each test centre
The graph gives the overall idea of the test performance based on pass rate and the
year.
As per the graph we can say that for year 2013, 2015, 2016, 2017 and 2018, the pass
rate is
higher that 55%. And the highest and lowest performance found in Kilkenny and
Monaghan test
centres respectively.

{r,echo=FALSE, fig.width=9,fig.height=4}

passfail$totPassPercentage <- round((passfail$Totalpass / (passfail$Totalpass +
passfail$totalFails)) * 100, digits = 2)
passfail$totFailPercentage <- round((passfail$totalFails / (passfail$Totalpass +
passfail$totalFails)) * 100, digits = 2)
#scatter plot for centre pass percetage per year
ggplot(data = passfail, aes(x = fct_reorder(Centre, -totPassPercentage), y =
totPassPercentage,
color = Year, size = totPassPercentage)) + geom_point(alpha = 0.5) +
theme(axis.text.x = element_text(size=9, angle=-90, hjust = 0, vjust = 0.5),
axis.ticks.x =
element_blank(), panel.background = element_rect(fill = "white", colour =
"lightblue"),
panel.grid.minor = element_line(size = 0.5, linetype = 'solid', colour = "lightblue")) +
labs(x = "Test Centres", y = "Total Pass %")#title = "Test centre pass% per year", #### Total pass count limits per year

The box plot shows the total pass count against each year. With this we can fetch the
details on
maximum and minimum pass counts per year, the meadian pass count and the
oustanding pass
count values which are shown as outliers (points) per year with the test centre

```
name.<br>
{r,echo=FALSE, fig.width=9,fig.height=4}
p <- plot_ly(passfail, x = passfail$Year, y = passfail$Totalpass, color =
~passfail$Year, type =
"box", text = paste("Centre = ", passfail$Centre)) %>%
layout(title = "Yearly performance", yaxis = list(title = 'Total Pass
Count'), xaxis =
list(title =
'Year'))
p
 Task_Rmd : ---
title: " Project – Signs with Smart Connectivity for Better Road Safety "
output: html_document ---
{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
#ggparcoord
#geom_polygon => packcircles
## Libraries
{r cars}
suppressMessages(library(readxl))
suppressMessages( library(dplyr))
suppressMessages( library(tidyr))
suppressMessages( library(ggplot2))
suppressMessages( library(MASS))
suppressMessages( library(GGally))
suppressMessages( library(ggExtra))
suppressMessages( library(plotly))
suppressMessages( library(packcircles))
 ## Preparing The Data For Analysis
```

```
{r}
df <- read_excel("mmAll.xlsx")
#d13 <- read_excel("m_m2013.xlsx")
#d14 <- read_excel("m_m2014.xlsx")
#d15 <- read_excel("m_m2015.xlsx")
#d16 <- read.csv("m_m2016.csv", header = T)
#d17 <- read_excel("m_m2017.xlsx")
#d18 <- read_excel("m_m2018.xlsx")
names(df)[9] <- "Vehicle and Safety Equipment"
names(df)[10] <- "Vehicle and Safety Equipment %"
names(df)[22] <- "Chassis and Body %"
names(df)[26] <- "Suspension Test %"
names(df)[36] <- "Incomplete Tests %"
df$reportYear <- as.factor(df$reportYear)
 ## Which part failed the most per report year?
{r, echo=FALSE}
######DATA#######
cols <- c("VehicleMake", "Vehicle and Safety Equipment", "Lighting and
Electrical",
"Steering
and Suspension", "Braking Equipment", "Wheels and Tyres", "Engine,
Noise and
Exhaust",
"Chassis and Body", "Side Slip Test", "Suspension Test", "Light test",
"Brake Test",
"Emmissions", "OTHER")
m <- df %>% dplyr :: select(c("reportYear", cols)) %>%
group_by(reportYear) %>%
summarise_if(is.numeric, mean, na.rm = TRUE)
```

```
m <- gather(m,-reportYear, key=Part, value= Failures)
{r}
######PLOT#######
#ggplot(m, aes(x=factor(reportYear), y=, colour=supp, group=supp)) +
geom_line()
library(MASS)
library(GGally)
# Vector color
library(RColorBrewer)
palette <- brewer.pal(3, "Set1")
my_colors <- palette[as.numeric(m$reportYear)]
#names(x) <- c("2013","2014","2015","2016","2017","2018")
#p <- ggparcoord(m, columns=2:13, groupColumn =
"reportYear")+geom_line(size=0.3)+theme_minimal() + geom_point()+
# xlab("Car Part")+ylab("Average failure rate")
ggplotly(ggplot(data=m, mapping = aes(x = reportYear, y = Failures,
colour = Part,
group=1))+
geom_point()+
geom_line()+xlab("Report Year")+ ylab("Average Number of Failures")
)
 ## Equipment Failures - Overall Statistics
{r}
library(ggplot2)
cols <- c("Vehicle and Safety Equipment", "Lighting and Electrical",
"Steering and
Suspension",
"Braking Equipment", "Wheels and Tyres", "Engine, Noise and Exhaust",
"Chassis
```

```
and Body",
"Side Slip Test", "Suspension Test", "Light test", "Brake Test",
"Emmissions",
"OTHER")
a <- df %>% dplyr::select(cols)
b<-colSums(a)
c <- data.frame(Part = names(b), Percent =
unname(b)/sum(df$Total)*100)
ggplot(c)+
geom_col(mapping = aes(x = reorder(Part, -Percent), y = Percent, fill =
Percent),
col="black")+
xlab("")+
ylab("Failure Percentage(%)") +
scale_fill_gradient(low = "orange", high = "tan")+
coord_flip()
 ### There is a bug in this code. Can anybody fix it?
{r, eval=FALSE}
####The polygon graph representation of the above data####
l <- data.frame(Part = names(b), Total = unname(b))
packing <- circleProgressiveLayout(l$Total,sizetype='area')l$packing <-
packing
l
dat.gg <- circleLayoutVertices(packing, npoints=50)
p <- ggplot() + geom_polygon(data = dat.gg, aes(x, y, group = id,
fill=as.factor(id)),
colour =
"black", alpha = 0.6) + geom_text(data = l, aes(x, y, size = Total, label =
Part))+scale_size_continuous(range = c(1,4)) +theme_void()
```

```
+theme(legend.position="none") +
coord_equal()
ggplotly(p, tooltip = c("Total", "Part"))

{r}
z <- df %>% group_by(VehicleMake) %>%
summarise(tot=sum(Total),res =
sum(PASS)/sum(Total)) %>% arrange(desc(tot)) %>% print(Inf())

 {r}
require(scales)
q <- z %>% arrange(desc(tot)) %>% slice(1:15)
ggplot(q)+
geom_col(mapping = aes(x = reorder(VehicleMake, -tot), y = tot, fill =
"green"))+
xlab("Vehicle Make")+ylab("Number of Vehicles") + coord_flip()+
theme(legend.position =
"none")+
scale_y_continuous(labels = comma)
#ggMarginal(g, type = "histogram", fill="transparent")
 ## Pass Percentage versus Number of Vehicles for a given VehicleMake
```{r}
require(scales)
library(plotly)
p <- ggplot(q, aes(x = tot, y = res*100))+
geom_line(color = "red")+
geom_point(aes(text = VehicleMake))+xlab("Number of Vehicles") +
ylab("Pass
Percentage
```
```

(%)") +

scale_x_continuous(labels = comma)

ggplotly(p, tooltip = "text")

1. Project main --- always_allow_html: yes output: html_document: default pdf_document: default --- <style type="text/css"> body{ /* Normal */ font-size: 12px; margin-left: 20px; }.column-left{ float: left; width: 40%; text-align: left; }.column-right{ float: right; width: 60%; text-align: right; }</style> {r setup, include=FALSE} knitr::opts_chunk$set(echo = TRUE) Date 15 Novemeber 2022 Team ID PNT2022TMID12612 Project Name Project – Signs with Smart Connectivity for Better Road Safetypackages <- c("plotly", "tidyverse", "ggmap", "GGally", "gridExtra", "scales", "viridis") newPackages <- packages[!(packages %in% installed.packages()[,"Package"])] if(length(newPackages)) install.packages(newPackages) library(tidyverse) library(plotly) library(gridExtra)

library(scales) library(GGally) library(viridis) library(ggmap) load("passfail.RData") {r,echo=FALSE} load("passfail.RData") passfail <- passfail %>% mutate(totalFails = Fail1 + ifelse(is.na(Fail2), 0, Fail2), Totalpass = Pass1 + ifelse(is.na(Pass2), 0, Pass2)) {r,echo=FALSE} passfailGroup <- summarise(group_by(passfail, Centre), Pass1 = sum(Pass1), Fail1 = sum(Fail1), Total1 = sum(Total1), Pass2 = sum(Pass2, na.rm = T), Fail2 = sum(Fail2, na.rm = T), Total2 = sum(Total2, na.rm = T), Totalpass = sum(Totalpass), totalFails = sum(totalFails)) passfailGroup <- mutate(passfailGroup, Pass1prop = Pass1/Total1, Pass2prop = Pass2/Total2, totalPassProp = (Totalpass / (Total1 + Total2)), totalFailsProp = (totalFails / (Total1 + Total2)))

{r,echo=FALSE} passfailGroup$totalPassProp = round((passfailGroup$totalPassProp * 100), digits = 2) passfailGroup$totalFailsProp = round((passfailGroup$totalFailsProp *

100), digits = 2) passFailGroup1 <- passfailGroup[c(1, 8)] passFailGroup1$Test <- "Pass" names(passFailGroup1) <- c("Centre", "Count", "Test") passFailGroup2 <- passfailGroup[c(1, 9)] passFailGroup2$Test <- "Fail" names(passFailGroup2) <- c("Centre", "Count", "Test") passFailcount <- rbind(passFailGroup1, passFailGroup2) ### Analysis based on test centres In this section we will analyse data from 2013 till 2018 about each test centre. As shown in the map , the test centres are spread across the Ireland and the number of centres is more in highly populated areas such as dublin, cork etc. The bar chart shows the total number of tests that each centre performed and the total pass and

fail counts as well as percentages. So, based on the test counts, the top 3 test centre are, Fonthill(770685), Deansgrade(767484), and Northpoint 2(729661). The botton 3 centres which performed less tests are, Donegal Town(16315), Cahirciveen(28806) and Clifden(38683). {r,echo=FALSE, fig.width=9,fig.height=3} t <- list(size = 8) p <- plot_ly(passfailGroup, x = ~passfailGroup$Centre, y = ~passfailGroup$Totalpass, type = 'bar', name = 'Pass', text = paste("Total tests = ", (passfailGroup$Totalpass+passfailGroup$totalFails), "
Passed =", passfailGroup$totalPassProp,"%", "
Failed =", passfailGroup$totalFailsProp,"%"), opacity = 0.5, marker = list(color = '#3AC3E3', line = list(color = '#0D6EB0', width = 1))) %>% add_trace(y = ~passfailGroup$totalFails, name = 'Fails', opacity = 0.5, marker = list(color = '#0E84FF', line = list(color = '#0D6EB0', width = 1))) %>% layout(yaxis = list(title = 'Count'), xaxis = list(title = 'Test Centres'), barmode = 'stack', font = t) p

<hr style = "margin: 10px 0px 10px;"> <div style = "display: inline-block;float: left;width: 50%;"> #### Total test passed for each test centre The following scatter plot show the total test pass count for each test centre from the year 2013 till year 2018. The questions that can be answered by this graph are,
.1. which are the top 3 and last 3 centres based on total pass count?
 (Deansgrade, Northpoint 2, Fonthill and Cahirciveen, Clifden, derrybeg resp.)
 2. Which year has the highest and lowest total pass count?
 2015 and 2014 respectively
 But, in this graph we are not considering the total tests performed by the test centres which shows the actual performance of the tests. For this we will plot another graph. </div> <div style = "display: inline-block;width: 50%;padding-left: 15px;margin-bottom: 90px;"> #### Test performance for each test centre The graph gives the overall idea of the test performance based on pass rate and the year.

As per the graph we can say that for year 2013, 2015, 2016, 2017 and 2018, the pass rate is higher that 55%. And the highest and lowest performance found in Kilkenny and Monaghan test centres respectively. </div> {r,echo=FALSE,include=T, fig.width=9,fig.height=3} #scatter plot for centre total pass per year passfail1 <- passfail passfail1$Centre <- fct_reorder(passfail1$Centre, - passfail1$Totalpass) passfail1$TotalPass1 <- passfail1$Totalpass p1 <- ggplotly(ggplot(data = passfail1, aes(x = Centre, y = Totalpass, color = Year, size = TotalPass1)) + geom_point(alpha = 0.5) + theme(axis.text.x = element_text(size=6, angle=-90, hjust = 0, vjust = 0.5), legend.position = "none", axis.ticks.x = element_blank(), panel.background = element_rect(fill = "white", colour = "lightblue"), panel.grid.major.y = element_line()) + labs(x = "Test Centres", y = "Totol pass count"), tooltip = c("Centre","Year", "Totalpass")) %>% layout(yaxis = list(gridcolor = toRGB("lightblue")), font = t)

<img src = "Result//3.jpg" style = "margin-left: 60px;margin- bottom: -18px;"> {r,echo=FALSE, fig.width=10,fig.height=3} passfail1$totPassPercentage <- round((passfail1$Totalpass / (passfail1$Totalpass + passfail1$totalFails)) * 100, digits = 2) passfail1$totFailPercentage <- round((passfail1$totalFails / (passfail1$Totalpass + passfail1$totalFails)) * 100, digits = 2) passfail1$totPassPercentage1 <- round((passfail1$Totalpass / (passfail1$Totalpass + passfail1$totalFails)) * 100, digits = 2) passfail1$Centre <- fct_reorder(passfail1$Centre, -passfail1$totPassPercentage) #scatter plot for centre pass percetage per year p2 <- ggplotly(ggplot(data = passfail1, aes(x = Centre, y = totPassPercentage, color = Year, size = totPassPercentage1)) + geom_point(alpha = 0.5) + theme(axis.text.x = element_text(size=6, angle=-90, hjust = 0, vjust = 0.5), legend.position = "none", legend.background = element_blank(), axis.ticks.x = element_blank(), panel.background = element_rect(fill = "white", colour = "lightblue"), panel.grid.minor = element_line(size = 0.5, linetype = 'solid', colour = "lightblue")) +
labs(x = "Test Centres", y = "Total Pass %"), tooltip = c("Centre","Year", "totPassPercentage")) %>% layout(yaxis = list(gridcolor = toRGB("lightblue")), font = t) #title = "Test centre pass% per year", <div style = "width: 100%;"> <div style = "float: left;display: flex;"> {r,echo=FALSE, fig.show="hold", fig.width=5, fig.height=3.5} p1 </div> <div style = "display: flex;"> {r,echo=FALSE, fig.show="hold", fig.width=5, fig.height=3.5} p2 </div></div> <hr style = "margin: 10px 0px 10px;"> <div style = "float: left;"> {r,echo=FALSE, fig.width=6,fig.height=2.5} p <- plot_ly(passfail, x = passfail$Year, y = passfail$Totalpass, color = ~passfail$Year, type = "box", text = paste("Centre = ", passfail$Centre)) %>%

layout(title = "Yearly performance", yaxis = list(title = 'Total Pass Count'), xaxis = list(title = 'Year'), showlegend = FALSE, font = t, legend = list(x = 0.9, y = 0.98)) p </div> <div style = "float: right;width: 35%;margin-top: 25px;"> #### Total pass count limits per year The box plot shows the total pass count against each year. With this we can fetch the details on maximum and minimum pass counts per year, the meadian pass count and the oustanding pass count values which are shown as outliers (points) per year with the test centre name. </div> 2.Road safety --title: "RoadSafety" author: "Amol | Haojun | Japneet | Calum" date: "11/11/2019" output: html_document --- {r setup, include=FALSE}

knitr::opts_chunk$set(echo = TRUE) {r} #Creating make n model data frame #reading excel file m_m <- readxl::read_excel('.\\DATA\\make_n_model\\mmAll.xlsx') #demo tag nrow(m_m) # m_m <- full_join(m_m13, m_m14) # paste(colnames(m_m13), " = ", colnames(m_m18)) 3. Final_projects Codes --- always_allow_html: yes author: "Amol | Haojun | Japneet | Calum" output: html_document: default pdf_document: default pagetitle: Road Safety --- <style type="text/css"> body{ /* Normal */ font-size: 12px; } .column-left{

float: left; width: 40%; text-align: left; }.column-right{ float: right; width: 60%; text-align: right; }/* Clear floats after the columns */ .row:after { content: ""; display: table; clear: both; }.column-left1{ float: left; width: 80%; text-align: left; }.column-right1{ float: right; width: 20%; text-align: right; padding-left: 15px;

padding-top: 15px; }.column-left2{ float: left; width: 47.5%; text-align: left; }.column-right2{ float: right; width: 47.5%; text-align: right; }</style> {r setup, include=FALSE} knitr::opts_chunk$set(echo = TRUE) ## Install and library necessary libraries packages <- c("plotly", "tidyverse", "ggmap", "GGally", "gridExtra", "scales", "viridis", "scatterplot3d", "readxl") newPackages <- packages[!(packages %in% installed.packages()[,"Package"])] if(length(newPackages)) install.packages(newPackages) library(tidyverse)library(plotly) library(gridExtra) library(scales)

library(GGally) library(viridis) library(ggmap) library(scatterplot3d) library(readxl) # Load necessary data files load("passfail.RData") load("nct_geom.RData") # Private API key for google maps. Please do not share. register_google("AIzaSyDy7z18GxhakN5ACVLsdqQfIm5B9jR mXpA") ## NCT Statistics Report {.tabset} ### Pass/Fail overview - Calum {r,echo=FALSE} # Data preparation passfailtotals <- summarise(group_by(passfail,Year),Pass1=sum(Pass1),Fail1=su m(Fail1),Total1=sum(Total1),Pass2=sum(Pass2),Fail2=sum(Fail2),Total 2=sum(Total2)) passfailtotals <- mutate(passfailtotals,Pass1prop=Pass1/Total1,Pass2prop=Pass2/ Total2)[c(1,2,3,4,8,5,6,7,9)] passfailtotals1 <- passfailtotals[c(1,2,3,4)] names(passfailtotals1) <- c("Year","Pass","Fail","Total") passfailtotals1$Test <- "First" passfailtotals2 <- passfailtotals[c(1,6,7,8)] names(passfailtotals2) <- c("Year","Pass","Fail","Total") passfailtotals2$Test <- "Retest" passfailtotals0 <- rbind(passfailtotals1,passfailtotals2) passfailtotals1 <- passfailtotals0[c(1,2,4,5)] names(passfailtotals1) <- c("Year","Count","Total","Test") passfailtotals1$Result <- "Pass" passfailtotals2 <- passfailtotals0[c(1,3,4,5)] names(passfailtotals2) <- c("Year","Count","Total","Test") passfailtotals2$Result <- "Fail" passfailtotals0 <- rbind(passfailtotals1,passfailtotals2) passfailtotals0$Result<- factor(passfailtotals0$Result,c("Pass","Fail")) passfailtotals0$Test<- factor(passfailtotals0$Test,c("First","Retest")) Let

us begin with an overview of the data. The NCT is a test that all cars over 4 years of age must undergo to legally drive on roads in Ireland. We have NCT pass and fail data for almost 12 million cars tested from 2013 to 2018. This data was recorded from all 47 test centres scattered across Ireland. This includes both initial test and retest data. Please note retest data was not available for 2014, hence it was
omitted from our report. Here's an overview of how this data is distributed. {r,echo=FALSE, warning=FALSE,fig.width=9, fig.height=2.5} # Pass/Fail count barplot p1 <- ggplot(passfailtotals0,aes(x=Year,y=Count, fill=Result))+ geom_col(position="dodge")+ theme_bw()+ theme(legend.position = "none",legend.title = element_blank())+ scale_fill_manual(values = c("lightblue","slategray"))+ facet_wrap(~Test)+ scale_y_continuous(labels = comma) # Pass/Fail rate barplot p2 <- ggplot(passfailtotals0,aes(x=Year,y=Count, fill=Result))+ geom_col(position="fill")+ labs(y="Proportion")+ geom_hline(yintercept = 0.5,col="red")+ theme_bw()+ theme(legend.key = element_rect(colour="black"), legend.position = c(0.912,0.85),legend.title = element_blank(), legend.background = element_rect(fill="transparent"), legend.text = element_text(size = 8))+ scale_fill_manual(values = c("lightblue","slategray"))+
facet_wrap(~Test) # Arrange plots side by side grid.arrange(p1, p2, ncol=2) As you can see the majority fail the first test, however the margins are quite close. As to be expected, the retest has a low fail rate.It is interesting to note that both total number of cars tested and pass proportion per year hasn't fluctuated much. One might expect that as the population increases, so too must the number of cars. One possible explanation for the lack of growth is that more people may be switching to public transport. We would also expect as technology advances cars

should become more reliable, yet our data does not support this theory. Perhaps the NCT have included stricter requirements that would balance this increase. <div class = "column-left">

 #### *Which test centre should I go to?* To the right we've ranked different centres by their first test pass proportions. Using an exponentially weighted mean we prioritized more recent results in our calculation. The top shows centres
with relatively high pass rates and the bottom shows the centres with the lowest. Notice how consistent the scores are. This could be dues to higher quality vehicles in more affluent areas or it could indicate a bias in the testing centres. Our recomendations are if you live in Monaghan, take a weekend trip to Kilkenny for your car test, you may end up saving money.

 #### *Is location a factor?* To test the above theory we created the map to the right. The colour represents the same scale as above, with size representing the total volume of cars in 2018. There is a large cluster of low ranking centres in north-central and north-west Ireland. This may support our affluency theory. If we look at the Dublin area there are low ranking centres to the north and higher ranking centres to the south. This could be a reflection of the northside - southside distribution of wealth. It is intriguing that Kerry has some of the highest ranked centres, despite being a more rural county. Traffic volume seems less significant there are large centres and small centres at either end of the spectrum. </div> <div class = "column-right">
{r,echo=FALSE, message=FALSE, include=F} # data preparation for parallel coords and map x <- data.frame(split(passfail$Pass1prop,passfail$Year)) names(x) <- c("2013","2014","2015","2016","2017","2018") x <- cbind(x,nct_geom) x$Total2018<- passfail$Total1[passfail$Year=="2018"] z <- rev(diff(c(0,pexp(1:6,0.5)))) x <-

arrange(x,desc(rowSums(mapply(`*`,select(x,starts_with("2")),z)))) x$Centre <-factor(x$Centre,levels=x$Centre) x$Rank <- 1:47 {r,echo=FALSE, message=FALSE, include=T} # Parallel coords plot p <- ggparcoord(x, columns=1:6, groupColumn = "Centre")+ geom_line(size=0.3)+ theme_minimal()+ scale_color_viridis(discrete = TRUE, direction = -1, option="C")+ labs(x="",y="") ggplotly(p, width = 550, height = 300, tooltip = c("Centre",".ID")) # Ireland map with data points Ire_map <- get_googlemap(center=c(-7.8,53.5), zoom=7,style =
'feature:administrative|element:labels|visibility:off') p <- ggmap(Ire_map)+ geom_point(data=x, aes(x=lat,y=lon, colour=Centre, size=Total2018))+ scale_radius(range=c(1,3))+ theme_bw()+ scale_color_viridis(discrete = TRUE, direction = - 1, option="C")+ theme(legend.position = "none")+ labs(x="", y="") ggplotly(p, width = 550, height = 300, tooltip=c("Centre","Total2018")) </div> ### Analysis based on test centres - Amol {r,echo=FALSE} load("passfail.RData") passfail <- passfail %>% mutate(totalFails = Fail1 + ifelse(is.na(Fail2), 0, Fail2), Totalpass = Pass1 + ifelse(is.na(Pass2), 0, Pass2)) {r,echo=FALSE} passfailGroup <- summarise(group_by(passfail, Centre), Pass1 = sum(Pass1), Fail1 = sum(Fail1),
Total1 = sum(Total1), Pass2 = sum(Pass2, na.rm = T), Fail2 = sum(Fail2, na.rm = T), Total2 = sum(Total2, na.rm = T), Totalpass = sum(Totalpass), totalFails = sum(totalFails)) passfailGroup <- mutate(passfailGroup, Pass1prop = Pass1/Total1, Pass2prop = Pass2/Total2, totalPassProp = (Totalpass / (Total1 + Total2)), totalFailsProp = (totalFails / (Total1 + Total2))) {r,echo=FALSE} passfailGroup$totalPassProp = round((passfailGroup$totalPassProp * 100), digits = 2) passfailGroup$totalFailsProp = round((passfailGroup$totalFailsProp * 100), digits = 2) passFailGroup1 <- passfailGroup[c(1, 8)] passFailGroup1$Test <- "Pass" names(passFailGroup1) <- c("Centre",

"Count", "Test") passFailGroup2 <- passfailGroup[c(1, 9)] passFailGroup2$Test <- "Fail" names(passFailGroup2) <- c("Centre", "Count", "Test") passFailcount <- rbind(passFailGroup1, passFailGroup2) In this section we will analyse data from 2013 till 2018 about each test centre. As shown in the map , the test centres are spread across the Ireland and the number of centres is more in highly populated areas such as dublin, cork etc. The bar chart shows the total number of tests that each centre performed and the total pass and fail counts as well as percentages. So, based on the test counts, the top 3 test centre are, Fonthill(770685), Deansgrade(767484), and Northpoint 2(729661). The botton 3 centres which performed less tests are, Donegal Town(16315), Cahirciveen(28806) and Clifden(38683). {r,echo=FALSE, fig.width=9,fig.height=2.8}t <- list(size = 8) p <- plot_ly(passfailGroup, x = ~passfailGroup$Centre, y = ~passfailGroup$Totalpass, type = 'bar', name = 'Pass', text = paste("Total tests = ", (passfailGroup$Totalpass+passfailGroup$totalFails), "
Passed =", passfailGroup$totalPassProp,"%", "
Failed =", passfailGroup$totalFailsProp,"%"), opacity = 0.5, marker = list(color = '#3AC3E3', line = list(color = '#0D6EB0', width = 1))) %>% add_trace(y = ~passfailGroup$totalFails, name = 'Fails', opacity = 0.5, marker = list(color = '#0E84FF', line = list(color = '#0D6EB0', width = 1))) %>% layout(yaxis = list(title = 'Count'), xaxis = list(title = 'Test Centres'), barmode = 'stack', font = t,legend = list(x = 0.93, y = 1)) p <hr style = "margin: 10px 0px 10px;"> <div style = "display: inline-block;float: left;width: 55%;"> #### Total test passed for each test centre The following scatter plot shows the total test pass count for each test centre from the year 2013 till year 2018. The questions that can

be answered by this graph are,
 1. which are the top 3 and last 3 centres based on total pass count?
 (Deansgrade, Northpoint 2, Fonthill and Cahirciveen, Clifden, derrybeg resp.)
 2. Which year has the highest and lowest total pass count?
 2015 and 2013 respectively
 But, in this graph we are not considering the total tests performed by the test centres which shows the actual performance of the tests. For this we will plot another graph.
</div> <div style = "display: inline-block;width: 45%;padding-left: 15px;margin-bottom: 30px;"> #### Test performance for each test centre The graph gives the overall idea of the test performance based on pass rate and the year. As per the graph we can say that for year 2013, 2015, 2016, 2017 and 2018, the pass rate is higher that 55%. And the highest and lowest performance found in Kilkenny and Monaghan test centres respectively. The case with the 2014 being less in number is because of the incomplete data available from the NCT website and it can be processed in the same manner if we have the complete set. </div> {r,echo=FALSE,include=F, fig.width=8.5,fig.height=3} #scatter plot for centre total pass per year passfail1 <- passfail passfail1$Centre <- fct_reorder(passfail1$Centre, - passfail1$Totalpass) passfail1$TotalPass1 <- passfail1$Totalpass p1 <- ggplotly(ggplot(data = passfail1, aes(x = Centre, y = Totalpass, color = Year, size = TotalPass1)) + geom_point(alpha = 0.5) +
theme(axis.text.x = element_text(size=6, angle=-90, hjust = 0, vjust = 0.5), legend.position = "none", axis.ticks.x = element_blank(), panel.background = element_rect(fill = "white", colour = "lightblue"), panel.grid.major.y = element_line()) + labs(x = "Test Centres", y = "Totol pass count"), tooltip = c("Centre","Year", "Totalpass")) %>% layout(yaxis = list(gridcolor = toRGB("lightblue")), font = t) <img src = "3.jpg" style = "margin-left: 60px;margin-bottom: - 10px;">

{r,echo=FALSE, fig.width=9,fig.height=3} passfail1$totPassPercentage <- round((passfail1$Totalpass / (passfail1$Totalpass + passfail1$totalFails)) * 100, digits = 2) passfail1$totFailPercentage <- round((passfail1$totalFails / (passfail1$Totalpass + passfail1$totalFails)) * 100, digits = 2) passfail1$totPassPercentage1 <- round((passfail1$Totalpass / (passfail1$Totalpass + passfail1$totalFails)) * 100, digits = 2) passfail1$Centre <- fct_reorder(passfail1$Centre, -passfail1$totPassPercentage) #scatter plot for centre pass percetage per year

p2 <- ggplotly(ggplot(data = passfail1, aes(x = Centre, y = totPassPercentage, color = Year, size = totPassPercentage1)) + geom_point(alpha = 0.5) + theme(axis.text.x = element_text(size=6, angle=-90, hjust = 0, vjust = 0.5), legend.position = "none", axis.ticks.x = element_blank(), panel.background = element_rect(fill = "white", colour = "lightblue"), panel.grid.minor = element_line(size = 0.5, linetype = 'solid', colour = "lightblue")) + labs(x = "Test Centres", y = "Total Pass %"), tooltip = c("Centre","Year", "totPassPercentage")) %>% layout(yaxis = list(gridcolor = toRGB("lightblue")), font = t) #title = "Test centre pass% per year", <div style = "width: 100%;"> <div style = "float: left;display: flex;"> {r,echo=FALSE, fig.show="hold", fig.width=4.75, fig.height=3.3} p1 </div> <div style = "display: flex;"> {r,echo=FALSE, fig.show="hold", fig.width=4.75, fig.height=3.3} p2 </div></div> <hr style = "margin: 10px 0px 10px;"> <div style = "float: left;"> {r,echo=FALSE, fig.width=6,fig.height=2.3} p <- plot_ly(passfail, x = passfail$Year, y = passfail$Totalpass, color = ~passfail$Year, type = "box", text = paste("Centre = ", passfail$Centre)) %>% layout(title = "Yearly performance", yaxis = list(title = 'Total Pass Count'), xaxis = list(title = 'Year'), font = t, legend = list(x = 0.92, y = 0.98, bgcolor = "transparent"), showlegend = FALSE) p </div> <div style = "float:

right;width: 35%;margin-top: 25px;"> #### Total pass count limits per year The box plot shows the total pass count against each year. With this we can fetch the details on maximum and minimum pass counts per year, the median pass count and the outstanding pass count values which are shown as outliers (points) per year with the test centre name. </div> ### Equipment Failure - Japneet{r, echo=FALSE}
####LOADING AND CLEANING THE DATASET#### df <- read_excel("mmAll.xlsx") names(df)[9] <- "Vehicle and Safety Equipment" names(df)[10] <- "Vehicle and Safety Equipment %" names(df)[22] <- "Chassis and Body %" names(df)[26] <- "Suspension Test %" names(df)[36] <- "Incomplete Tests %" df$reportYear <- as.factor(df$reportYear) <div class = "column-left">

 #### *Equipment Failure - An Overview*

 <p style="text-align:justify"> The barplot, resulting from the exploratory data analysis, arranges the different vehicle item categories in decreasing order of their failure percentage over a span of 6 years altogether. Overall, Lighting and Electrical is the most failed item category with a failure percentage of 19.87 whereas Body and Chassis being the least failed known category with a failure percentage of 4.67. The category Other being the least failed item category,
overall, includes the parts that are not covered in the major 12 categories and hence is the area of least interest for this analysis. </p>

 #### *Analyzing Part Failures Per Report Year*

 <p style="text-align:justify">Diving further, we derive interesting insights on analyzing the item failure for each report year. Among the top 3 failure items overall, the Lighting and Electrical holds the topmost position throughout the entire span with a fail percentage hovering just around 20. However, the failure percentage for Steering and Suspension follows an increasing trend from 2014 to 2018 with an

increase of 2.089%, which moves it up the list from third position in 2014 to a second position in 2015. A corresponding decrease in failure percentage of wheels and

Tyres is observed which moves it down the list to become the third most failed item in 2018. </p>

 #### *Is there Any Relationship between Top Vehicle Makes and Top 3 item failure categories?*

 <p style="text-align:justify"> Certainly Yes. TOYOTA seem to have the lowest failure percentage among all the vehicle makes for all the three item categories. Collectively, all the top 5 makes have improved their 'Wheels and Tyres' over the 6 report years. However, an increase in failure percentage for 'Light and Electrical' and 'Steering and Suspension' is observed for almost all the makes with NISSAN and VOLKSWAGEN being an exception with a slight decrease of 0.348% for NISSAN and that of 2.154% for VOLKSWAGEN in failure percentage of 'Light and

Electrical' parts. </p> </div> <div class = "column-right">

 <p style="text-align:center"> {r, echo=FALSE, fig.height=5, fig.width=12, warning=FALSE} #########PLOT 1########## cols <- c("Vehicle and Safety Equipment", "Lighting and Electrical", "Steering and Suspension", "Braking Equipment", "Wheels and Tyres", "Engine, Noise and Exhaust", "Chassis and Body", "Side Slip Test", "Suspension Test", "Light test", "Brake Test", "Emmissions", "OTHER") a <- df %>% dplyr::select(cols) b<-colSums(a) c <- data.frame(Part = names(b), Percent = unname(b)/sum(df$Total)*100)p1 <- ggplot(c)+ geom_col(mapping = aes(x = reorder(Part, -Percent), y = Percent, fill = Percent), col="black")+ xlab("")+ theme_light()+ ylab("Failure Percentage(%)") +

scale_fill_gradient(low = "lightblue", high = "brown")+ coord_flip()+ scale_y_continuous(labels = function(x) paste0(x, "%"))+

theme(legend.position = "none", panel.grid.major.x = element_blank(), panel.border = element_blank(), panel.grid.major.y = element_blank(), panel.grid.minor = element_blank(), axis.text=element_text(size=16), axis.title = element_text(size = 20))+ geom_text(aes(x = Part, y = Percent+0.9, label = round(Percent, 2)), size = 5) p1

 {r, echo=FALSE, warning=FALSE} #########PLOT 2########## cols <- c("Total", "Vehicle and Safety Equipment", "Lighting and Electrical", "Steering and Suspension", "Braking Equipment", "Wheels and Tyres", "Engine, Noise and Exhaust", "Chassis and Body", "Side Slip Test", "Suspension Test", "Light test", "Brake Test", "Emmissions", "OTHER") s <- df %>% dplyr :: select(c("reportYear", cols)) %>% group_by(reportYear) %>%

summarise_if(is.numeric, sum, na.rm = TRUE) %>% mutate_at(vars(c(-1,-2)), funs(round((. / Total)*100, digits = 3))) m <- gather(s,-reportYear, key=Part, value= Failures) m <- m[7:84,] p2 <- ggplotly(ggplot(data=m, mapping = aes(x = reportYear, y = Failures, colour = Part, group=1))+ geom_point()+ theme_minimal()+ geom_line()+xlab("Report Year")+ ylab("Failure Percentage") + scale_y_continuous(labels = function(x) paste0(x, "%")), height=400) p2 {r, echo=FALSE, warning=FALSE, message=FALSE,fig.height=4, fig.width=5.5}#########PLOT 3########## e <- c("VehicleMake", "reportYear", "Total", "Lighting and Electrical", "Steering and Suspension", "Wheels and Tyres") s <- df %>% dplyr::select(e) %>% filter(VehicleMake %in% c("TOYOTA", "VOLKSWAGEN", "FORD", "NISSAN", "OPEL")) %>% group_by(VehicleMake, reportYear) %>% summarise_if(is.numeric, sum, na.rm = TRUE) %>%

mutate_at(vars(c(-1,-2, -3)), funs(round((. / Total)*100, digits = 3))) plot_ly(x=s$`Lighting and Electrical`, y=s$`Steering and Suspension`, z=s$`Wheels and Tyres`, type="scatter3d", mode="lines", color=

as.factor(s$VehicleMake), marker = list(symbol = 'circle', sizemode = 'diameter'), sizes = c(5, 150), text= s$reportYear, hovertemplate = paste('<i>Report Year</i>: %{text}', '
Lighting and Electrical: %{x}%', '
Steering and Suspension: %{y}%', '
Wheels and Tyres: %{z}%')) %>% layout(scene = list(xaxis = list(title = 'Lighting and Electrical (%)'), yaxis = list(title = 'Steering and Suspension (%)'), zaxis = list(title = 'Wheels and Tyres (%)'))) </p> </div> ### Make/Model analysis - Haojun {r echo=FALSE } ################### raw data ################### mmdata<-read_excel("mmAll1.xlsx",sheet=2,na="NA") {r ,echo=FALSE} ################### select top 15 market share make's name ################### # totol number of car in each make TotalMumMake <- mmdata %>% group_by(VehicleMake) %>% summarise(MakeTotal=sum(Total, na.rm=T)) # top 15 make names TotalMumMake<-arrange(TotalMumMake,desc(MakeTotal)) Name15<-TotalMumMake$VehicleMake[1:15] {r , echo=FALSE} ################### prepare data for market share plot ################### # the number of car of top 15 make in each year TotaMumlMakeYear <- mmdata %>% group_by(VehicleMake,reportYear) %>% summarise(MakeTotal=sum(Total, na.rm=T))# the number of car in each year TotalMumYear<- mmdata %>% group_by(reportYear) %>% summarise(YearTotal=sum(Total, na.rm=T)) # left join TotaMumlMakeYear and TotalMumYear MarketShare<-left_join(TotaMumlMakeYear,TotalMumYear,by="reportYear") # calculate market share of each brand in each year
MarketShare$marketshare<-
MarketShare$MakeTotal/MarketShare$YearTotal # select market share of top 15 make Top15<-filter(MarketShare,VehicleMake %in%

c("TOYOTA",'VOLKSWAGEN' ,'FORD','NISSAN','OPEL', 'RENAULT','PEUGEOT','BMW','AUDI','MERCEDES BENZ', 'HYUNDAI','SKODA','HONDA','MAZDA','CITROEN')) Top15<- arrange(Top15,desc(marketshare)) #sort make for plot Top15$VehicleMake <- factor(Top15$VehicleMake, levels=c('TOYOTA', 'VOLKSWAGEN', 'FORD', 'NISSAN', 'OPEL', 'RENAULT','PEUGEOT','BMW','AUDI','MERCEDES BENZ','HYUNDAI','SKODA', 'HONDA','MAZDA','CITROEN'), ordered=TRUE) <div class = "column-left1" style = "width: 65%;"> #### *Market Share of Car Makes* {r Top15,echo=FALSE,fig.height=5,fig.width=10} ################## plot of market share for top 15 make ################## ggplot()+ geom_point(data=Top15,mapping = aes(x=marketshare,y= VehicleMake,color=reportYear),size=2)+ geom_line(data=Top15,mapping = aes(x=marketshare ,y=VehicleMake,color=reportYear),size=2)+ labs(x="Market Share",y="Vehicle Make", color="Year")+ theme_bw()+ theme(axis.text.x = element_text(face="bold", color="black", size=14,angle = 45,vjust = 0.6), axis.text.y = element_text(face="bold", color="black", size=14), plot.title = element_text(hjust = 0.5), axis.title = element_text(size = 17), legend.position = c(0.9,0.6), legend.text = element_text(size=14), legend.title = element_blank()) + coord_flip()+ scale_x_continuous(labels = scales::percent,breaks=seq(0,1,0.05)) {r ,echo=FALSE} ####### data for plot of fail rate and distribution of each car age cut ####### # car age mmdata_1<- cbind(mmdata,CarAge=mmdata$reportYear- mmdata$YearOfBirth) # cut car age mmdata_1$CarAge_cut<-cut(mmdata_1$CarAge,c(-999,4,6,8,10,12,14,16,18,999),labels = c(

'[0,4]','(4,6]','(6,8]','(8,10]','(10,12]','(12,14]','(14,16]','(16,18]','(1 8,+)')) # summarize number group by CarAge_cut TotalMumAge <- mmdata_1 %>% group_by(CarAge_cut) %>% summarise(MakeTotal=sum(Total, na.rm=T), FailTotal=sum(FAIL, na.rm=T)) # calculate fail rate for each age cut TotalMumAge$FailRate<-
TotalMumAge$FailTotal/TotalMumAge$MakeTotal # distribution of car age TotalMumAge$CarAgePer<-
TotalMumAge$MakeTotal/sum(TotalMumAge$MakeTotal)
 #### *Fail Rate and Proportion of Car Ages* {r ,echo=FALSE,fig.height=4,fig.width=10} ################ plot of fail rate and distribution of each car age group ################ cols <- c("Fail Rate" = "red", "Proportion" = "skyblue") ggplot(data = TotalMumAge) + geom_point(mapping = aes(x = CarAge_cut, y = FailRate,colour="Fail Rate"),size=3)+ labs(x="Car Age",y="")+
geom_bar(aes(CarAge_cut,weight=CarAgePer,fill="Proportion") ,colour ="black",width = 0.5) + theme_bw()+ theme(plot.title = element_text(hjust = 0.5), axis.text.x = element_text(face="bold", color="black", size=14), axis.text.y = element_text(face="bold", color="black",size=14), legend.position = c(0.9,0.6), legend.text = element_text(size=14), axis.title = element_text(size = 17))+ scale_colour_manual(name = "",values=cols)+ scale_fill_manual(name="",values=cols)+ scale_y_continuous(labels = scales::percent)

 </div> <div class = "column-right1" style = "text-align: left;width: 35%;"> <p>

 The plot shows the top 15 market share of car makes,which occupy more than 85% of the total market
in Ireland. Toyota, Volkswagen and Ford rank top 3 market share in recent six years and have kept stable.The market share of Nissan, Opel

and Renault have declined significantly from 2013 to 2018.But for Audi,Hyundai and Skoda, it has increased gradually.

 The bar chart is the distribution of car ages, which shows the largest proportion of cars are between 8 and 14 years old. The point above the bar represents the fail rate of cars of this age in the first test. As the car age increase, fail rate rises linearly. </p>

 </div> {r ,echo=FALSE} ################ plot of fail rate for top 15 makes in different age cut ################## # summarize number for different makes in different age group TotalMumAgeMake <- mmdata_1 %>% group_by(VehicleMake,CarAge_cut) %>% summarise(MakeTotal=sum(Total, na.rm=T), FailTotal=sum(FAIL, na.rm=T)) # calculate fail rate TotalMumAgeMake$FailRate<- TotalMumAgeMake$FailTotal/TotalMumAgeMake$MakeTotal # selsect top 15 makes for plot TotalMumAgeMake<- filter(TotalMumAgeMake,VehicleMake %in% c("TOYOTA",'VOLKSWAGEN' ,'FORD','NISSAN','OPEL','RENAULT ','PEUGEOT','BMW','A UDI','MERCEDES BENZ', 'HYUNDAI','SKODA','HONDA','MAZDA' ,'CITROEN'))
calculate 1st Qu., median and 3rd Qu. of fail rate MakeFailRate<- summary(subset(TotalMumAgeMake,TotalMumAgeMake$Car Age_cut=="(10,12]")$FailRate) # sort makes for plot lables TotalMumAgeMake$VehicleMake <- factor(TotalMumAgeMake$VehicleMake, levels=c('HONDA','TOYOTA','MAZDA','NISSAN','MERCED ES BENZ','FORD', 'VOLKSWAGEN','OPEL','SKODA','BMW','AUDI','CITROEN', 'PEUGEOT','HYUNDAI','RENAULT'))

{r,echo=FALSE,fig.height=3,fig.width=5 } ################ plot of fail ratefor top 15 makes in different age cut ##################p1<-ggplot(data = TotalMumAgeMake,mapping = aes(y = FailRate, x=VehicleMake)) + geom_boxplot(fill="lightgoldenrod")+ geom_hline(yintercept =c(0.5536,0.6139),colour="red",linetype="dotted")+ geom_hline(yintercept =c(0.5962),colour="red")+ geom_text(aes(x=-0.2,y=0.5536,label ="LQ",hjust=-0.2, vjust = 0.9), size = 2)+ geom_text(aes(x=-0.2,y=0.5962,label ="MED",hjust=-0.1, vjust = 0.9), size = 2)+ geom_text(aes(x=-0.2,y=0.6139,label ="UQ",hjust=-0.1, vjust = -0.2), size = 2)+ labs(x="Vehicle Make", y="Fail Rate")+ ggtitle("Fail Rate of Car Makes")+ theme_bw()+ theme(axis.text.x = element_text(angle = 45, face="bold", color="black", size=8,vjust = 0.6), axis.text.y = element_text(face="bold", color="black", size=8), plot.title = element_text(hjust = 0.5,vjust = 0))+ scale_y_continuous(labels = scales::percent) {r,echo=FALSE } ################# select quality of top 5 and bottom 5 model ################### ## choose Top 15 brand mmdata_2<-filter(mmdata_1,VehicleMake %in% c("TOYOTA",'VOLKSWAGEN' ,'FORD','NISSAN','OPEL', 'RENAULT','PEUGEOT','BMW','AUDI','MERCEDES BENZ','HYUNDAI','SKODA','HONDA','MAZDA' ,'CITROEN'))# compare the fail rate of different model in in same age cut((10,12]) goodMakeModel <- subset(mmdata_2,CarAge_cut=="(10,12]") %>% group_by(VehicleMake,VehicleModel) %>% summarise(MakeModelTotal=sum(Total, na.rm=T), FailTotal=sum(FAIL, na.rm=T)) #calculate distribution of model goodMakeModel$MakeModelPercent<-goodMakeModel$MakeModelTotal/sum(goodMakeModel$Mak eModelTotal) # calculate fial rate of model goodMakeModel$MakeModelFialRate<- goodMakeModel$FailTotal/goodMakeModel$MakeModelTotal # sort percentage of model goodMakeModel<- arrange(goodMakeModel,desc(MakeModelPercent)) # choose popular model as the range of analysis (the number of these models shoold occupy at lease 80% of total) # choose top 50 model as range of analysis (the number of these models is at least 85% of total) #sum(goodMakeModel[1:50,]$MakeModelPercent) Top5Model<- arrange(goodMakeModel[1:50,],goodMakeModel[1:50,]$Make ModelFialRate) # choose quality top 5 and bottom 5 model for plot BestWorst5Model<-Top5Model[c(1:5,46:50),] BestWorst5Model$MakeModel<- paste(BestWorst5Model$VehicleMake,BestWorst5Model$Vehi cleModel,sep = "-") # sort model for plot BestWorst5Model$MakeModel<- factor(BestWorst5Model$MakeModel, levels=c("TOYOTA YARIS","TOYOTA-RAV 4","TOYOTA-COROLLA","HONDA-CIVIC","FORD-FIESTA", "RENAULT SCENIC", "RENAULT-MEGANE","FORD- GALAXY","RENAULT-LAGUNA","HYUNDAI TRAJET"))BestWorst5Model$rank<- ifelse(BestWorst5Model$MakeModelFialRate>0.6,"Bottom 5 Model","Top 5 Model") #### *Car Makes and Models Recommended* {r,echo=FALSE, ,fig.height=3,fig.width=9} p2<-ggplot() + geom_bar(BestWorst5Model,mapping=aes(x=MakeModel,weig ht=MakeModelFialRate,fill=rank), width=0.5, colour="black")+ scale_fill_manual(values = c("firebrick1","seagreen1"))+ labs(x="Car Model",y="Fail Rate")+ ggtitle("Top and Bottom 5 Model")+ theme_bw()+ theme(axis.text.x = element_text(face="bold", color="black", size=8,angle = 45, vjust = 0.6), axis.text.y =

element_text(face="bold", color="black", size=8), plot.title = element_text(hjust = 0.5,vjust=0), legend.position = c(0.18,0.8), legend.title = element_blank(), legend.background = element_rect(fill="transparent"))+ scale_y_continuous(labels = scales::percent) grid.arrange(p1, p2, ncol=2) <div class = "column-left2"> The box plot shows fail rates of different car ages of vehicle makes. The red lines are upper quartile, median and lower quartile fail rate in different car age groups of vehicle makes. Honda, Toyota and Mazda have a better quality, but Renault, Hyundai, Peugeot and Citroen are easy to fail in test. </div> <div class = "column-right2" style = "text-align: left;"> The bar chart shows top 5 and bottom 5 models in fail rates. These models are selected from most popular 50 models of car between 10 and 12 years old, which occupy more than 85% market share in Ireland. TOYOTA-YARIS,TOYOTA-RAV 4,TOYOTA-COROLLA,HONDA-CIVIC and FORD FIESTA are the most recommended models. Potential buyers of RENAULT-SCENIC,RENAULT MEGANE,FORD-GALAXY,RENAULT-LAGUNA and HYUNDAI-TRAJET should be aware they have the least reliable rates. </div> 4.Label Name The label are attached to github. https://github.com/IBM-EPBL/IBM-Project-9461-1659009035/tree/main/PROJECT-DEVELOPMENT 5. Regarding Thanks # Thanks - Check out [https://github.com/IBM-EPBL/IBM-Project-9461-1659009035.](https://github.com/IBM-EPBL/IBM-Project-9461-1659009035)

SPRINT 4:

---
always_allow_html: yes
output:
html_document: default
pdf_document: default

````
---
<style type="text/css">
body{ /* Normal */
font-size: 12px;
margin-left: 20px;
}
.column-left{
float: left;
width: 40%;
text-align: left;
}
.column-right{
float: right;
width: 60%;
text-align: right;
}
</style>
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
Date
15 NOVEMBER 2022
Team ID
PNT2022TMID12612
Project Name
Project – Signs with Smart Connectivity for Better Road Safetypackages
<- c("plotly", "tidyverse", "ggmap", "GGally", "gridExtra", "scales",
"viridis")
newPackages <- packages[!(packages %in%
installed.packages()[,"Package"])]
if(length(newPackages)) install.packages(newPackages)
library(tidyverse)
library(plotly)
library(gridExtra)
library(scales)
library(GGally)
library(viridis)
library(ggmap)
load("passfail.RData")
```
```{r,echo=FALSE}
load("passfail.RData")
passfail <- passfail %>%
````
````

mutate(totalFails = Fail1 + ifelse(is.na(Fail2), 0, Fail2), Totalpass = Pass1
+ ifelse(is.na(Pass2),
0, Pass2))
```
```{r,echo=FALSE}
passfailGroup <- summarise(group_by(passfail, Centre), Pass1 =
sum(Pass1), Fail1 = sum(Fail1),
Total1 = sum(Total1), Pass2 = sum(Pass2, na.rm = T), Fail2 = sum(Fail2,
na.rm = T), Total2 =
sum(Total2, na.rm = T), Totalpass = sum(Totalpass), totalFails =
sum(totalFails))
passfailGroup <- mutate(passfailGroup, Pass1prop = Pass1/Total1,
Pass2prop = Pass2/Total2,
totalPassProp = (Totalpass / (Total1 + Total2)), totalFailsProp =
(totalFails / (Total1 + Total2)))
```
```{r,echo=FALSE}
passfailGroup$totalPassProp = round((passfailGroup$totalPassProp *
100), digits = 2)
passfailGroup$totalFailsProp = round((passfailGroup$totalFailsProp *
100), digits = 2)
passFailGroup1 <- passfailGroup[c(1, 8)]
passFailGroup1$Test <- "Pass"
names(passFailGroup1) <- c("Centre", "Count", "Test")
passFailGroup2 <- passfailGroup[c(1, 9)]
passFailGroup2$Test <- "Fail"
names(passFailGroup2) <- c("Centre", "Count", "Test")
passFailcount <- rbind(passFailGroup1, passFailGroup2)
```
Analysis based on test centres
In this section we will analyse data from 2013 till 2018 about each test centre. As shown in the <a
href = "https://github.com/NanawareAmol/R-project_Road safety/blob/master/Result/loc_spread_across_ireland.JPG">map, the test centres are spread
across the Ireland and the number of centres is more in highly populated areas such as dublin,
cork etc.
The bar chart shows the total number of tests that each centre performed and the total pass and
fail counts as well as percentages. So, based on the test counts, the top 3 test centre are, *Fonthill(770685)*, *Deansgrade(767484)*, and *Northpoint 2(729661)*. The botton 3 centres
```

which performed less tests are, *Donegal Town(16315)*,
Cahirciveen(28806) and
Clifden(38683).
```{r,echo=FALSE, fig.width=9,fig.height=3}
t <- list(size = 8)
p <- plot_ly(passfailGroup, x = ~passfailGroup$Centre, y =
~passfailGroup$Totalpass, type =
'bar', name = 'Pass', text = paste("Total tests = ",
(passfailGroup$Totalpass+passfailGroup$totalFails), "<br>Passed =",
passfailGroup$totalPassProp,"%", "<br>Failed =",
passfailGroup$totalFailsProp,"%"), opacity =
0.5, marker = list(color = '#3AC3E3', line = list(color = '#0D6EB0', width
= 1))) %>%
add_trace(y = ~passfailGroup$totalFails, name = 'Fails', opacity = 0.5,
marker = list(color = '#0E84FF', line = list(color = '#0D6EB0', width =
1))) %>%
layout(yaxis = list(title = 'Count'), xaxis = list(title = 'Test Centres'),
barmode = 'stack', font = t)
p
```
<hr style = "margin: 10px 0px 10px;">
<div style = "display: inline-block;float: left;width: 50%;">
Total test passed for each test centre
The following scatter plot show the total test pass count for each test centre from the year 2013
till year 2018. The questions that can be answered by this graph are,

1. which are the top 3 and last 3 centres based on total pass count?
    (Deansgrade, Northpoint 2, Fonthill and
Cahirciveen, Clifden, derrybeg resp.)
2. Which year has the highest and lowest total pass count?
    2015 and 2014 respectively
But, in this graph we are not considering the total tests performed by the test centres which shows
the actual performance of the tests. For this we will plot another graph.
</div>
<div style = "display: inline-block;width: 50%;padding-left: 15px;margin-bottom: 90px;">
Test performance for each test centre
The graph gives the overall idea of the test performance based on pass rate and the year.
As per the graph we can say that for year 2013, 2015, 2016, 2017 and 2018, the pass rate is higher

that 55%. And the highest and lowest performance found in Kilkenny and
Monaghan test centres
respectively.

````
</div>
```{r,echo=FALSE,include=T, fig.width=9,fig.height=3}
#scatter plot for centre total pass per year
passfail1 <- passfail
passfail1$Centre <- fct_reorder(passfail1$Centre, -passfail1$Totalpass)
passfail1$TotalPass1 <- passfail1$Totalpass
p1 <- ggplotly(ggplot(data = passfail1, aes(x = Centre, y = Totalpass,
color = Year, size =
TotalPass1)) + geom_point(alpha = 0.5) +
theme(axis.text.x = element_text(size=6, angle=-90, hjust = 0, vjust =
0.5), legend.position =
"none", axis.ticks.x = element_blank(), panel.background =
element_rect(fill = "white", colour =
"lightblue"), panel.grid.major.y = element_line()) +
labs(x = "Test Centres", y = "Totol pass count"), tooltip =
c("Centre","Year", "Totalpass"))
%>% layout(yaxis = list(gridcolor = toRGB("lightblue")), font = t)```
<img src = "Result//3.jpg" style = "margin-left: 60px;margin-bottom: -
18px;">
```{r,echo=FALSE, fig.width=10,fig.height=3}
passfail1$totPassPercentage <- round((passfail1$Totalpass /
(passfail1$Totalpass +
passfail1$totalFails)) * 100, digits = 2)
passfail1$totFailPercentage <- round((passfail1$totalFails /
(passfail1$Totalpass +
passfail1$totalFails)) * 100, digits = 2)
passfail1$totPassPercentage1 <- round((passfail1$Totalpass /
(passfail1$Totalpass +
passfail1$totalFails)) * 100, digits = 2)
passfail1$Centre <- fct_reorder(passfail1$Centre, -
passfail1$totPassPercentage)
#scatter plot for centre pass percetage per year
p2 <- ggplotly(ggplot(data = passfail1, aes(x = Centre, y =
totPassPercentage, color = Year, size
= totPassPercentage1)) + geom_point(alpha = 0.5) +
theme(axis.text.x = element_text(size=6, angle=-90, hjust = 0, vjust =
0.5), legend.position =
"none", legend.background = element_blank(), axis.ticks.x =
element_blank(), panel.background
= element_rect(fill = "white", colour = "lightblue"),
````

````
panel.grid.minor = element_line(size = 0.5, linetype = 'solid', colour =
"lightblue")) +
labs(x = "Test Centres", y = "Total Pass %"), tooltip = c("Centre","Year",
"totPassPercentage"))
%>% layout(yaxis = list(gridcolor = toRGB("lightblue")), font = t) #title
= "Test centre pass%
per year",
```
<div style = "width: 100%;">
<div style = "float: left;display: flex;">
```{r,echo=FALSE, fig.show="hold", fig.width=5, fig.height=3.5}
p1
```
</div>
<div style = "display: flex;">
```{r,echo=FALSE, fig.show="hold", fig.width=5, fig.height=3.5}
p2
```
</div></div>
<hr style = "margin: 10px 0px 10px;">
<div style = "float: left;">
```{r,echo=FALSE, fig.width=6,fig.height=2.5}
p <- plot_ly(passfail, x = passfail$Year, y = passfail$Totalpass, color =
~passfail$Year, type =
"box", text = paste("Centre = ", passfail$Centre)) %>%
layout(title = "Yearly performance", yaxis = list(title = 'Total Pass
Count'), xaxis = list(title =
'Year'), showlegend = FALSE, font = t, legend = list(x = 0.9, y = 0.98))
p
```
</div>
<div style = "float: right;width: 35%;margin-top: 25px;">
#### <b>Total pass count limits per year</b>
The box plot shows the total pass count against each year. With this we
can fetch the details on
maximum and minimum pass counts per year, the meadian pass count
and the oustanding pass
count values which are shown as outliers (points) per year with the test
centre name.
</div>
````

2.Road safety

--

title: "RoadSafety"
````

~~~~
author: "Amol | Haojun | Japneet | Calum"
date: "11/11/2019"
output: html_document
---
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
```
```{r}
#Creating make n model data frame
#reading excel file
m_m <- readxl::read_excel('.\\DATA\\make_n_model\\mmAll.xlsx')
#demo tag
nrow(m_m)
# m_m <- full_join(m_m13, m_m14)
# paste(colnames(m_m13), " = ", colnames(m_m18))
```
~~~~

3. Final_projects Codes

~~~~
---
always_allow_html: yes author: "Amol | Haojun | Japneet | Calum"
output:
html_document: default
pdf_document: default
pagetitle: Road Safety
---
<style type="text/css">
body{ /* Normal */
font-size: 12px; }
.column-left{
float: left;
width: 40%;
text-align: left;
}
.column-right{
float: right;
width: 60%;
text-align: right;
}
/* Clear floats after the columns */
.row:after {
content: "";
display: table;
clear: both;
}
~~~~

```
.column-left1{
float: left;
width: 80%;
text-align: left;
}
.column-right1{
float: right;
width: 20%;
text-align: right;
padding-left: 15px;
padding-top: 15px;
}
.column-left2{
float: left;
width: 47.5%;
text-align: left;
}
.column-right2{
float: right;
width: 47.5%;
text-align: right;
}
</style>
```{r setup, include=FALSE}
knitr::opts_chunk$set(echo = TRUE)
## Install and library necessary libraries
packages <- c("plotly", "tidyverse", "ggmap", "GGally", "gridExtra",
"scales", "viridis", "scatterplot3d",
"readxl")
newPackages <- packages[!(packages %in%
installed.packages()[,"Package"])]
if(length(newPackages)) install.packages(newPackages)
library(tidyverse)library(plotly)
library(gridExtra)
library(scales)
library(GGally)
library(viridis)
library(ggmap)
library(scatterplot3d)
library(readxl)
# Load necessary data files
load("passfail.RData")
load("nct_geom.RData")
```
```

````
# Private API key for google maps. Please do not share.
register_google("AIzaSyDy7z18GxhakN5ACVLsdqQfIm5B9jRmXpA")
```
## NCT Statistics Report {.tabset}
### Pass/Fail overview - Calum
```{r,echo=FALSE}
# Data preparation
passfailtotals <-
summarise(group_by(passfail,Year),Pass1=sum(Pass1),Fail1=sum(Fail1),
Total1=sum(Total1),Pass2=sum(Pass2
),Fail2=sum(Fail2),Total2=sum(Total2))
passfailtotals <-
mutate(passfailtotals,Pass1prop=Pass1/Total1,Pass2prop=Pass2/Total2)[c
(1,2,3,4,8,5,6,7,9)]
passfailtotals1 <- passfailtotals[c(1,2,3,4)]
names(passfailtotals1) <- c("Year","Pass","Fail","Total")
passfailtotals1$Test <- "First"
passfailtotals2 <- passfailtotals[c(1,6,7,8)]
names(passfailtotals2) <- c("Year","Pass","Fail","Total")
passfailtotals2$Test <- "Retest"
passfailtotals0 <- rbind(passfailtotals1,passfailtotals2)
passfailtotals1 <- passfailtotals0[c(1,2,4,5)]
names(passfailtotals1) <- c("Year","Count","Total","Test")
passfailtotals1$Result <- "Pass"
passfailtotals2 <- passfailtotals0[c(1,3,4,5)]
names(passfailtotals2) <- c("Year","Count","Total","Test")
passfailtotals2$Result <- "Fail"
passfailtotals0 <- rbind(passfailtotals1,passfailtotals2)
passfailtotals0$Result<-factor(passfailtotals0$Result,c("Pass","Fail"))
passfailtotals0$Test<-factor(passfailtotals0$Test,c("First","Retest"))
```
````

Let us begin with an overview of the data. The NCT is a test that all cars over 4 years of age must undergo to legally drive on roads in Ireland. We have NCT pass and fail data for almost 12 million cars tested from 2013 to 2018. This data was recorded from all 47 test centres scattered across Ireland. This
includes both initial test and retest data. Please note retest data was not available for 2014, hence it was
omitted from our report. Here's an overview of how this data is distributed.

````
```{r,echo=FALSE, warning=FALSE,fig.width=9, fig.height=2.5}
# Pass/Fail count barplot
p1 <- ggplot(passfailtotals0,aes(x=Year,y=Count, fill=Result))+
````

```
geom_col(position="dodge")+
theme_bw()+
theme(legend.position = "none",legend.title = element_blank())+
scale_fill_manual(values = c("lightblue","slategray"))+
facet_wrap(~Test)+
scale_y_continuous(labels = comma)
# Pass/Fail rate barplot
p2 <- ggplot(passfailtotals0,aes(x=Year,y=Count, fill=Result))+
geom_col(position="fill")+
labs(y="Proportion")+
geom_hline(yintercept = 0.5,col="red")+
theme_bw()+
theme(legend.key = element_rect(colour="black"), legend.position =
c(0.912,0.85),legend.title =
element_blank(), legend.background = element_rect(fill="transparent"),
legend.text = element_text(size = 8))+
scale_fill_manual(values = c("lightblue","slategray"))+
facet_wrap(~Test)
# Arrange plots side by side
grid.arrange(p1, p2, ncol=2 )
```

As you can see the majority fail the first test, however the margins are quite close. As to be expected, the retest has a low fail rate.It is interesting to note that both total number of cars tested and pass proportion per year hasn't fluctuated much. One might expect that as the population increases, so too must the number of cars. One possible explanation for the lack of growth is that more people may be switching to public transport. We would also expect as technology advances cars should become more reliable, yet our data does not support this theory. Perhaps the NCT have included stricter requirements that would balance this increase.

<div class = "column-left">

Which test centre should I go to?

To the right we've ranked different centres by their first test pass proportions. Using an exponentially weighted mean we prioritized more recent results in our calculation. The top shows centres with relatively high pass rates and the bottom shows the centres with the lowest. Notice how consistent the scores are. This could be dues to

higher quality vehicles in more affluent areas or it could indicate a bias in the testing centres. Our recomendations are if you live in Monaghan, take a weekend trip to Kilkenny for your car test, you may end up saving money.

Is location a factor?

To test the above theory we created the map to the right. The colour represents the same scale as above, with size representing the total volume of cars in 2018. There is a large cluster of low ranking centres in north-central and north-west Ireland. This may support our affluency theory. If we look at the Dublin area there are low ranking centres to the north and higher ranking centres to the south. This could be a reflection of the northside - southside distribution of wealth. It is intriguing that Kerry has some of the highest ranked centres, despite being a more rural county. Traffic volume seems less significant there are large centres and small centres at either end of the spectrum.

</div>

<div class = "column-right">

````
```{r,echo=FALSE, message=FALSE, include=F}
# data preparation for parallel coords and map
x <- data.frame(split(passfail$Pass1prop,passfail$Year))
names(x) <- c("2013","2014","2015","2016","2017","2018")
x <- cbind(x,nct_geom)
x$Total2018<- passfail$Total1[passfail$Year=="2018"]
z <- rev(diff(c(0,pexp(1:6,0.5))))
x <- arrange(x,desc(rowSums(mapply(`*`,select(x,starts_with("2")),z))))
x$Centre <-factor(x$Centre,levels=x$Centre)
x$Rank <- 1:47
```
```{r,echo=FALSE, message=FALSE, include=T}
# Parallel coords plot
p <- ggparcoord(x, columns=1:6, groupColumn = "Centre")+
geom_line(size=0.3)+
theme_minimal()+
scale_color_viridis(discrete = TRUE, direction = -1, option="C")+
labs(x="",y="")
ggplotly(p, width = 550, height = 300, tooltip = c("Centre",".ID"))
# Ireland map with data points
````
````

````
Ire_map <- get_googlemap(center=c(-7.8,53.5), zoom=7,style =
'feature:administrative|element:labels|visibility:off')
p <- ggmap(Ire_map)+
geom_point(data=x, aes(x=lat,y=lon, colour=Centre, size=Total2018))+
scale_radius(range=c(1,3))+
theme_bw()+ scale_color_viridis(discrete = TRUE, direction = -1,
option="C")+
theme(legend.position = "none")+
labs(x="", y="")
ggplotly(p, width = 550, height = 300, tooltip=c("Centre","Total2018"))
```
</div>
### Analysis based on test centres - Amol
```{r,echo=FALSE}
load("passfail.RData")
passfail <- passfail %>%
mutate(totalFails = Fail1 + ifelse(is.na(Fail2), 0, Fail2), Totalpass = Pass1
+ ifelse(is.na(Pass2), 0,
Pass2))
```
```{r,echo=FALSE}
passfailGroup <- summarise(group_by(passfail, Centre), Pass1 =
sum(Pass1), Fail1 = sum(Fail1),
Total1 = sum(Total1), Pass2 = sum(Pass2, na.rm = T), Fail2 = sum(Fail2,
na.rm = T), Total2 = sum(Total2,
na.rm = T), Totalpass = sum(Totalpass), totalFails = sum(totalFails))
passfailGroup <- mutate(passfailGroup, Pass1prop = Pass1/Total1,
Pass2prop = Pass2/Total2,
totalPassProp = (Totalpass / (Total1 + Total2)), totalFailsProp =
(totalFails / (Total1 + Total2)))
```
```{r,echo=FALSE}
passfailGroup$totalPassProp = round((passfailGroup$totalPassProp *
100), digits = 2)
passfailGroup$totalFailsProp = round((passfailGroup$totalFailsProp *
100), digits = 2)
passFailGroup1 <- passfailGroup[c(1, 8)]
passFailGroup1$Test <- "Pass"
names(passFailGroup1) <- c("Centre", "Count", "Test")
passFailGroup2 <- passfailGroup[c(1, 9)]
passFailGroup2$Test <- "Fail"
names(passFailGroup2) <- c("Centre", "Count", "Test")
passFailcount <- rbind(passFailGroup1, passFailGroup2)
````

````
```
In this section we will analyse data from 2013 till 2018 about each test
centre. As shown in the <a href =
"https://github.com/NanawareAmol/R-project_Road
safety/blob/master/Result/loc_spread_across_ireland.JPG">map</a>, the
test centres are spread across the
Ireland and the number of centres is more in highly populated areas such
as dublin, cork etc.
The bar chart shows the total number of tests that each centre performed
and the total pass and fail
counts as well as percentages. So, based on the test counts, the top 3 test
centre are, *Fonthill(770685)*,
*Deansgrade(767484)*, and *Northpoint 2(729661)*. The botton 3
centres which performed less tests are,
*Donegal Town(16315)*, *Cahirciveen(28806)* and *Clifden(38683)*.
```{r,echo=FALSE, fig.width=9,fig.height=2.8}t <- list(size = 8)
p <- plot_ly(passfailGroup, x = ~passfailGroup$Centre, y =
~passfailGroup$Totalpass, type = 'bar',
name = 'Pass', text = paste("Total tests = ",
(passfailGroup$Totalpass+passfailGroup$totalFails), "<br>Passed
=", passfailGroup$totalPassProp,"%", "<br>Failed =",
passfailGroup$totalFailsProp,"%"), opacity = 0.5, marker
= list(color = '#3AC3E3', line = list(color = '#0D6EB0', width =
1))) %>%
add_trace(y = ~passfailGroup$totalFails, name = 'Fails', opacity = 0.5,
marker = list(color = '#0E84FF', line = list(color = '#0D6EB0', width =
1))) %>%
layout(yaxis = list(title = 'Count'), xaxis = list(title = 'Test Centres'),
barmode = 'stack', font = t,legend
= list(x = 0.93, y = 1))
p
```
<hr style = "margin: 10px 0px 10px;">
<div style = "display: inline-block;float: left;width: 55%;">
#### <b>Total test passed for each test centre</b>
The following scatter plot shows the total test pass count for each test
centre from the year 2013
till year 2018. The questions that can be answered by this graph are,
<br/>
1. which are the top 3 and last 3 centres based on total pass
count?<br/>
    <b>(Deansgrade, Northpoint 2, Fonthill and
Cahirciveen, Clifden, derrybeg resp.)</b><br/>
````
````

**2. Which year has the highest and lowest total pass count?
**
**2015 and 2013 respectively
**
But, in this graph we are not considering the total tests performed by the test centres which shows
the actual performance of the tests. For this we will plot another graph.
</div>
<div style = "display: inline-block;width: 45%;padding-left: 15px;margin-bottom: 30px;">
Test performance for each test centre
The graph gives the overall idea of the test performance based on pass rate and the year.
As per the graph we can say that for year 2013, 2015, 2016, 2017 and 2018, the pass rate is higher that
55%. And the highest and lowest performance found in Kilkenny and Monaghan test centres respectively. The
case with the 2014 being less in number is because of the incomplete data available from the NCT website and it
can be processed in the same manner if we have the complete set.
</div>
```{r,echo=FALSE,include=F, fig.width=8.5,fig.height=3}
#scatter plot for centre total pass per year
passfail1 <- passfail
passfail1$Centre <- fct_reorder(passfail1$Centre, -passfail1$Totalpass)
passfail1$TotalPass1 <- passfail1$Totalpass
p1 <- ggplotly(ggplot(data = passfail1, aes(x = Centre, y = Totalpass,
color = Year, size = TotalPass1))
+ geom_point(alpha = 0.5) +
theme(axis.text.x = element_text(size=6, angle=-90, hjust = 0, vjust =
0.5), legend.position = "none",
axis.ticks.x = element_blank(), panel.background = element_rect(fill =
"white", colour = "lightblue"),
panel.grid.major.y = element_line()) + labs(x = "Test Centres", y = "Totol
pass count"), tooltip = c("Centre","Year", "Totalpass")) %>%
layout(yaxis = list(gridcolor = toRGB("lightblue")), font = t)
```
<img src = "3.jpg" style = "margin-left: 60px;margin-bottom: -10px;">
```{r,echo=FALSE, fig.width=9,fig.height=3}
passfail1$totPassPercentage <- round((passfail1$Totalpass /
(passfail1$Totalpass +
passfail1$totalFails)) * 100, digits = 2)
passfail1$totFailPercentage <- round((passfail1$totalFails /
(passfail1$Totalpass + passfail1$totalFails))
```

````
* 100, digits = 2)
passfail1$totPassPercentage1 <- round((passfail1$Totalpass /
(passfail1$Totalpass +
passfail1$totalFails)) * 100, digits = 2)
passfail1$Centre <- fct_reorder(passfail1$Centre, -
passfail1$totPassPercentage)
#scatter plot for centre pass percetage per year
p2 <- ggplotly(ggplot(data = passfail1, aes(x = Centre, y =
totPassPercentage, color = Year, size =
totPassPercentage1)) + geom_point(alpha = 0.5) +
theme(axis.text.x = element_text(size=6, angle=-90, hjust = 0, vjust =
0.5), legend.position = "none",
axis.ticks.x = element_blank(), panel.background = element_rect(fill =
"white", colour = "lightblue"),
panel.grid.minor = element_line(size = 0.5, linetype = 'solid', colour =
"lightblue")) +
labs(x = "Test Centres", y = "Total Pass %"), tooltip = c("Centre","Year",
"totPassPercentage")) %>%
layout(yaxis = list(gridcolor = toRGB("lightblue")), font = t) #title =
"Test centre pass% per year",
```
<div style = "width: 100%;">
<div style = "float: left;display: flex;">
```{r,echo=FALSE, fig.show="hold", fig.width=4.75, fig.height=3.3}
p1
```
</div>
<div style = "display: flex;">
```{r,echo=FALSE, fig.show="hold", fig.width=4.75, fig.height=3.3}
p2
```
</div></div>
<hr style = "margin: 10px 0px 10px;">
<div style = "float: left;">
```{r,echo=FALSE, fig.width=6,fig.height=2.3}
p <- plot_ly(passfail, x = passfail$Year, y = passfail$Totalpass, color =
~passfail$Year, type = "box",
text = paste("Centre = ", passfail$Centre)) %>%
layout(title = "Yearly performance", yaxis = list(title = 'Total Pass
Count'), xaxis = list(title = 'Year'),
font = t, legend = list(x = 0.92, y = 0.98, bgcolor = "transparent"),
showlegend = FALSE)
p
````

~~~~
```
</div>
<div style = "float: right;width: 35%;margin-top: 25px;">
#### <b>Total pass count limits per year</b>
The box plot shows the total pass count against each year. With this we
can fetch the details on
maximum and minimum pass counts per year, the median pass count and
the outstanding pass count values
which are shown as outliers (points) per year with the test centre name.
</div>
### Equipment Failure - Japneet```{r, echo=FALSE}
####LOADING AND CLEANING THE DATASET####
df <- read_excel("mmAll.xlsx")
names(df)[9] <- "Vehicle and Safety Equipment"
names(df)[10] <- "Vehicle and Safety Equipment %"
names(df)[22] <- "Chassis and Body %"
names(df)[26] <- "Suspension Test %"
names(df)[36] <- "Incomplete Tests %"
df$reportYear <- as.factor(df$reportYear)
```
<div class = "column-left">
<br>
<br>
#### **Equipment Failure - An Overview**
<br>
<br>
<p style="text-align:justify">
The barplot, resulting from the exploratory data analysis, arranges the
different vehicle item categories
in decreasing order of their failure percentage over a span of 6 years
altogether. Overall, Lighting and Electrical
is the most failed item category with a failure percentage of 19.87
whereas Body and Chassis being the least
failed known category with a failure percentage of 4.67. The category
Other being the least failed item category,
overall, includes the parts that are not covered in the major 12 categories
and hence is the area of least interest
for this analysis.
</p>
<br>
<br>
<br>
<br>
~~~~


Analyzing Part Failures Per Report Year

<p style="text-align:justify">Diving further, we derive interesting insights on analyzing the item failure for each report year. Among the top 3 failure items overall, the Lighting and Electrical holds the topmost position throughout the entire span
with a fail percentage hovering just around 20. However, the failure percentage for Steering and Suspension
follows an increasing trend from 2014 to 2018 with an increase of 2.089%, which moves it up the list from third
position in 2014 to a second position in 2015. A corresponding decrease in failure percentage of wheels and
Tyres is observed which moves it down the list to become the third most failed item in 2018.
</p>

Is there Any Relationship between Top Vehicle Makes and Top 3 item failure categories?

<p style="text-align:justify">
Certainly Yes. TOYOTA seem to have the lowest failure percentage among all the vehicle makes for all
the three item categories. Collectively, all the top 5 makes have improved their 'Wheels and Tyres' over the 6
report years. However, an increase in failure percentage for 'Light and Electrical' and 'Steering and Suspension'
is observed for almost all the makes with NISSAN and VOLKSWAGEN being an exception with a slight
decrease of 0.348% for NISSAN and that of 2.154% for VOLKSWAGEN in failure percentage of 'Light and
Electrical' parts.
</p>
</div>

```
<div class = "column-right">
<br>
<br>
<br>
<p style="text-align:center">
```{r, echo=FALSE, fig.height=5, fig.width=12, warning=FALSE}
#########PLOT 1##########
cols <- c("Vehicle and Safety Equipment", "Lighting and Electrical",
"Steering and Suspension",
"Braking Equipment", "Wheels and Tyres", "Engine, Noise and Exhaust",
"Chassis and Body", "Side Slip
Test", "Suspension Test", "Light test", "Brake Test", "Emmissions",
"OTHER")
a <- df %>% dplyr::select(cols)
b<-colSums(a)
c <- data.frame(Part = names(b), Percent =
unname(b)/sum(df$Total)*100)p1 <- ggplot(c)+
geom_col(mapping = aes(x = reorder(Part, -Percent), y = Percent, fill =
Percent), col="black")+
xlab("")+ theme_light()+
ylab("Failure Percentage(%)") +
scale_fill_gradient(low = "lightblue", high = "brown")+
coord_flip()+ scale_y_continuous(labels = function(x) paste0(x, "%"))+
theme(legend.position = "none", panel.grid.major.x = element_blank(),
panel.border =
element_blank(), panel.grid.major.y = element_blank(), panel.grid.minor
= element_blank(),
axis.text=element_text(size=16), axis.title = element_text(size = 20))+
geom_text(aes( x = Part, y = Percent+0.9, label = round(Percent, 2)), size
= 5)
p1
```



```{r, echo=FALSE, warning=FALSE}
#########PLOT 2##########
cols <- c("Total", "Vehicle and Safety Equipment", "Lighting and
Electrical", "Steering and
Suspension", "Braking Equipment", "Wheels and Tyres", "Engine, Noise
and Exhaust", "Chassis and Body",
"Side Slip Test", "Suspension Test", "Light test", "Brake Test",
"Emmissions", "OTHER")
```
```

~~~
s <- df %>% dplyr :: select(c("reportYear", cols)) %>%
group_by(reportYear) %>%
summarise_if(is.numeric, sum, na.rm = TRUE) %>% mutate_at(vars(c(-
1,-2)), funs(round((. / Total)*100,
digits = 3)))
m <- gather(s,-reportYear, key=Part, value= Failures)
m <- m[7:84, ]
p2 <- ggplotly(ggplot(data=m, mapping = aes(x = reportYear, y =
Failures, colour = Part, group=1))+
geom_point()+ theme_minimal()+
geom_line()+xlab("Report Year")+ ylab("Failure Percentage") +
scale_y_continuous(labels =
function(x) paste0(x, "%")), height=400)
p2
```
```{r, echo=FALSE, warning=FALSE, message=FALSE,fig.height=4,
fig.width=5.5}#########PLOT 3##########
e <- c("VehicleMake", "reportYear", "Total", "Lighting and Electrical",
"Steering and Suspension",
"Wheels and Tyres")
s <- df %>% dplyr::select(e) %>%
filter(VehicleMake %in% c("TOYOTA", "VOLKSWAGEN", "FORD",
"NISSAN", "OPEL")) %>%
group_by(VehicleMake, reportYear) %>% summarise_if(is.numeric, sum,
na.rm = TRUE) %>%
mutate_at(vars(c(-1,-2, -3)), funs(round((. / Total)*100, digits = 3)))
plot_ly(x=s$`Lighting and Electrical`, y=s$`Steering and Suspension`,
z=s$`Wheels and Tyres`,
type="scatter3d", mode="lines", color= as.factor(s$VehicleMake),
marker = list(symbol = 'circle', sizemode =
'diameter'), sizes = c(5, 150), text= s$reportYear, hovertemplate =
paste('<i>Report Year</i>: %{text}',
'<br><b>Lighting and Electrical</b>: %{x}%',
'<br><b>Steering and Suspension</b>: %{y}%',
'<br><b>Wheels and Tyres</b>: %{z}%')) %>%
layout(scene = list(xaxis = list(title = 'Lighting and Electrical (%)'),
yaxis = list(title = 'Steering and Suspension (%)'),
zaxis = list(title = 'Wheels and Tyres (%)')))
```
</p>
</div>
### Make/Model analysis - Haojun
```{r echo=FALSE }
~~~

```
################### raw data ###################
mmdata<-read_excel("mmAll1.xlsx",sheet=2,na="NA")
```
```{r ,echo=FALSE}
################### select top 15 market share make's name
###################
# totol number of car in each make
TotalMumMake <- mmdata %>%
group_by(VehicleMake) %>%
summarise( MakeTotal=sum(Total, na.rm=T))
# top 15 make names
TotalMumMake<-arrange(TotalMumMake,desc(MakeTotal))
Name15<-TotalMumMake$VehicleMake[1:15]
```
```{r , echo=FALSE}
################### prepare data for market share plot
###################
# the number of car of top 15 make in each year
TotaMumlMakeYear <- mmdata %>%
group_by(VehicleMake,reportYear) %>%
summarise( MakeTotal=sum(Total, na.rm=T))# the number of car in each
year
TotalMumYear<- mmdata %>%
group_by(reportYear) %>%
summarise( YearTotal=sum(Total, na.rm=T))
# left join TotaMumlMakeYear and TotalMumYear
MarketShare<-
left_join(TotaMumlMakeYear,TotalMumYear,by="reportYear")
# calculate market share of each brand in each year
MarketShare$marketshare<-
MarketShare$MakeTotal/MarketShare$YearTotal
# select market share of top 15 make
Top15<-filter(MarketShare,VehicleMake %in%
c( "TOYOTA",'VOLKSWAGEN'
,'FORD','NISSAN','OPEL',
'RENAULT','PEUGEOT','BMW','AUDI','MERCEDES BENZ',
'HYUNDAI','SKODA','HONDA','MAZDA','CITROEN'))
Top15<-arrange(Top15,desc(marketshare))
#sort make for plot
Top15$VehicleMake <- factor(Top15$VehicleMake,
levels=c('TOYOTA', 'VOLKSWAGEN', 'FORD',
```

````
'NISSAN', 'OPEL',
'RENAULT','PEUGEOT','BMW','AUDI','MERCEDES
BENZ','HYUNDAI','SKODA',
'HONDA','MAZDA','CITROEN'), ordered=TRUE)
```
<div class = "column-left1" style = "width: 65%;">
#### **Market Share of Car Makes**
```{r Top15,echo=FALSE,fig.height=5,fig.width=10}
################### plot of market share for top 15 make
###################
ggplot()+
geom_point(data=Top15,mapping = aes(x=marketshare,y=
VehicleMake,color=reportYear),size=2)+
geom_line(data=Top15,mapping =
aes(x=marketshare ,y=VehicleMake,color=reportYear),size=2)+
labs(x="Market Share",y="Vehicle Make", color="Year")+
theme_bw()+
theme(axis.text.x = element_text(face="bold", color="black",
size=14,angle = 45,vjust = 0.6),
axis.text.y = element_text(face="bold", color="black", size=14),
plot.title = element_text(hjust = 0.5),
axis.title = element_text(size = 17),
legend.position = c(0.9,0.6),
legend.text = element_text(size=14),
legend.title = element_blank()) +
coord_flip()+
scale_x_continuous(labels = scales::percent,breaks=seq(0,1,0.05))
```
```{r ,echo=FALSE}
####### data for plot of fail rate and distribution of each car age cut
#######
# car age
mmdata_1<-cbind(mmdata,CarAge=mmdata$reportYear-
mmdata$YearOfBirth)
# cut car age
mmdata_1$CarAge_cut<-cut(mmdata_1$CarAge,c(-
999,4,6,8,10,12,14,16,18,999),labels = c(
'[0,4]','(4,6]','(6,8]','(8,10]','(10,12]','(12,14]','(14,16]','(16,18]','(18,+)'))
# summarize number group by CarAge_cut
TotalMumAge <- mmdata_1 %>%
group_by(CarAge_cut) %>% summarise( MakeTotal=sum(Total,
na.rm=T),
FailTotal=sum(FAIL, na.rm=T) )
````

```
# calculate fail rate for each age cut
TotalMumAge$FailRate<-
TotalMumAge$FailTotal/TotalMumAge$MakeTotal
# distribution of car age
TotalMumAge$CarAgePer<-
TotalMumAge$MakeTotal/sum(TotalMumAge$MakeTotal)
```
```
<br>
#### **Fail Rate and Proportion of Car Ages**
```{r ,echo=FALSE,fig.height=4,fig.width=10}
################# plot of fail rate and distribution of each car age group
#################
cols <- c("Fail Rate" = "red", "Proportion" = "skyblue")
ggplot(data = TotalMumAge) +
geom_point(mapping = aes(x = CarAge_cut, y = FailRate,colour="Fail
Rate"),size=3)+
labs(x="Car Age",y="")+
geom_bar(aes(CarAge_cut,weight=CarAgePer,fill="Proportion"),colour=
"black",width = 0.5) +
theme_bw()+
theme(plot.title = element_text(hjust = 0.5),
axis.text.x = element_text( face="bold", color="black", size=14),
axis.text.y = element_text( face="bold", color="black",size=14),
legend.position = c(0.9,0.6),
legend.text = element_text(size=14),
axis.title = element_text(size = 17))+
scale_colour_manual(name = "",values=cols)+
scale_fill_manual(name="",values=cols)+
scale_y_continuous(labels = scales::percent)
```


</div>
<div class = "column-right1" style = "text-align: left;width: 35%;">
<p>

The plot shows the top 15 market share of car makes,which occupy more
than 85% of the total market
in Ireland. Toyota, Volkswagen and Ford rank top 3 market share in
recent six years and have kept stable.The
```

market share of Nissan, Opel and Renault have declined significantly from 2013 to 2018.But for Audi,Hyundai
and Skoda, it has increased gradually.

The bar chart is the distribution of car ages, which shows the largest proportion of cars are between 8
and 14 years old. The point above the bar represents the fail rate of cars of this age in the first test. As the car
age increase, fail rate rises linearly.
</p>

</div>

````
```{r ,echo=FALSE}
################# plot of fail rate for top 15 makes in different age cut
###################
# summarize number for different makes in different age group
TotalMumAgeMake <- mmdata_1 %>%
group_by(VehicleMake,CarAge_cut) %>%
summarise( MakeTotal=sum(Total, na.rm=T),
FailTotal=sum(FAIL, na.rm=T))
# calculate fail rate
TotalMumAgeMake$FailRate<-
TotalMumAgeMake$FailTotal/TotalMumAgeMake$MakeTotal
# selsect top 15 makes for plot
TotalMumAgeMake<-filter(TotalMumAgeMake,VehicleMake %in%
c( "TOYOTA",'VOLKSWAGEN'
````
````

,'FORD','NISSAN','OPEL','RENAULT','PEUGEOT','BMW','AUDI','ME
RCEDES BENZ',
'HYUNDAI','SKODA','HONDA','MAZDA' ,'CITROEN'
))
calculate 1st Qu., median and 3rd Qu. of fail rate
MakeFailRate<-
summary(subset(TotalMumAgeMake,TotalMumAgeMake$CarAge_cut=
="(10,12]")$FailRate)
sort makes for plot lables
TotalMumAgeMake$VehicleMake <-
factor(TotalMumAgeMake$VehicleMake,
levels=c('HONDA','TOYOTA','MAZDA','NISSAN','MERCEDES
BENZ','FORD',
'VOLKSWAGEN','OPEL','SKODA','BMW','AUDI','CITROEN','PEUGE
OT','HYUNDAI','RENAULT'))
```
```{r,echo=FALSE,fig.height=3,fig.width=5 }
################# plot of fail ratefor top 15 makes in different age cut
###################p1<-ggplot(data = TotalMumAgeMake,mapping =
aes(y = FailRate, x=VehicleMake)) +
geom_boxplot(fill="lightgoldenrod")+
geom_hline(yintercept
=c(0.5536,0.6139),colour="red",linetype="dotted")+
geom_hline(yintercept =c(0.5962),colour="red")+
geom_text(aes(x=-0.2,y=0.5536,label ="LQ",hjust=-0.2, vjust = 0.9), size
= 2)+
geom_text(aes(x=-0.2,y=0.5962,label ="MED",hjust=-0.1, vjust = 0.9),
size = 2)+
geom_text(aes(x=-0.2,y=0.6139,label ="UQ",hjust=-0.1, vjust = -0.2),
size = 2)+
labs(x="Vehicle Make", y="Fail Rate")+
ggtitle("Fail Rate of Car Makes")+
theme_bw()+
theme(axis.text.x = element_text(angle = 45, face="bold", color="black",
size=8,vjust = 0.6),
axis.text.y = element_text(face="bold", color="black", size=8),
plot.title = element_text(hjust = 0.5,vjust = 0))+
scale_y_continuous(labels = scales::percent)
```
```{r,echo=FALSE }
################# select quality of top 5 and bottom 5 model
###################
## choose Top 15 brand

```
mmdata_2<-filter(mmdata_1,VehicleMake %in%
c( "TOYOTA",'VOLKSWAGEN'
,'FORD','NISSAN','OPEL',
'RENAULT','PEUGEOT','BMW','AUDI','MERCEDES
BENZ','HYUNDAI','SKODA','HONDA','MAZDA' ,'CITROEN'
))
# compare the fail rate of different model in in same age cut((10,12])
goodMakeModel <- subset(mmdata_2,CarAge_cut=="(10,12]") %>%
group_by(VehicleMake,VehicleModel) %>%
summarise( MakeModelTotal=sum(Total, na.rm=T),
FailTotal=sum(FAIL, na.rm=T))
#calculate distribution of model
goodMakeModel$MakeModelPercent<-
goodMakeModel$MakeModelTotal/sum(goodMakeModel$MakeModelT
otal)
# calculate fial rate of model
goodMakeModel$MakeModelFialRate<-
goodMakeModel$FailTotal/goodMakeModel$MakeModelTotal
# sort percentage of model
goodMakeModel<-arrange(goodMakeModel,desc(MakeModelPercent))
# choose popular model as the range of analysis (the number of these
models shoold occupy at lease
80% of total)
# choose top 50 model as range of analysis (the number of these models
is at least 85% of total)
#sum(goodMakeModel[1:50,]$MakeModelPercent)
Top5Model<-
arrange(goodMakeModel[1:50,],goodMakeModel[1:50,]$MakeModelFia
lRate)
# choose quality top 5 and bottom 5 model for plot
BestWorst5Model<-Top5Model[c(1:5,46:50),]
BestWorst5Model$MakeModel<-
paste(BestWorst5Model$VehicleMake,BestWorst5Model$VehicleModel,
sep = "-")
# sort model for plot
BestWorst5Model$MakeModel<- factor(BestWorst5Model$MakeModel,
levels=c("TOYOTA
YARIS","TOYOTA-RAV 4","TOYOTA-COROLLA","HONDA-
CIVIC","FORD-FIESTA", "RENAULT
SCENIC", "RENAULT-MEGANE","FORD-GALAXY","RENAULT-
LAGUNA","HYUNDAI
TRAJET"
))
```

BestWorst5Model$rank<-
ifelse(BestWorst5Model$MakeModelFialRate>0.6,"Bottom 5
Model","Top 5
Model")
```
#### **Car Makes and Models Recommended**
```{r,echo=FALSE, ,fig.height=3,fig.width=9}
p2<-ggplot() +
geom_bar(BestWorst5Model,mapping=aes(x=MakeModel,weight=Make
ModelFialRate,fill=rank),
width=0.5, colour="black")+
scale_fill_manual(values = c("firebrick1","seagreen1"))+
labs(x="Car Model",y="Fail Rate")+
ggtitle("Top and Bottom 5 Model")+
theme_bw()+
theme(axis.text.x = element_text(face="bold", color="black",
size=8,angle = 45, vjust = 0.6),
axis.text.y = element_text(face="bold", color="black", size=8),
plot.title = element_text(hjust = 0.5,vjust=0),
legend.position = c(0.18,0.8),
legend.title = element_blank(),
legend.background = element_rect(fill="transparent"))+
scale_y_continuous(labels = scales::percent)
grid.arrange(p1, p2, ncol=2 )
```
<div class = "column-left2">
The box plot shows fail rates of different car ages of vehicle makes. The red lines are upper quartile,
median and lower quartile fail rate in different car age groups of vehicle makes. Honda, Toyota and Mazda have
a better quality, but Renault, Hyundai, Peugeot and Citroen are easy to fail in test.
</div>
<div class = "column-right2" style = "text-align: left;">
The bar chart shows top 5 and bottom 5 models in fail rates. These models are selected from most
popular 50 models of car between 10 and 12 years old, which occupy more than 85% market share in
Ireland. TOYOTA-YARIS,TOYOTA-RAV 4,TOYOTA-COROLLA,HONDA-CIVIC and FORD
FIESTA are the most recommended models. Potential buyers of RENAULT-SCENIC,RENAULT
```

MEGANE,FORD-GALAXY,RENAULT-LAGUNA and HYUNDAI-TRAJET should be aware they have the least reliable rates.
</div>

CHAPTER 8

8. TESTING

8.1 Test Cases

- **TEST CASE 1**

Clear weather - Usual Speed Limit.

- **TEST CASE 2**

Foggy Weather - Reduced Speed Limit.

- **TEST CASE 3**

Rainy Weather - Further Reduced Speed Limit.

- **TEST CASE 4**

School/Hospital Zone - Do not Honk sign is displayed.

8.2 User Acceptance Testing

Dynamic speed & diversion variations based on the weather and traffic helps user to avoid traffic and have a safe journey home. The users would welcome this idea to be implemented everywhere.

CHAPTER 9

9. RESULTS

9.1 Performance Metrics

Based on the IBM pack we chose, the performance of the website varies. Built upon NodeJS, a light and high performance engine, Node RED is capable of handling up to 10,000 requests per second. Moreover, since the system is horizontally scalable, an even higher demand of customers can be served.

CHAPTER 10

10. ADVANTAGES & DISADVANTAGES

ADVANTAGES

▪ Lower battery consumption since processing is done mostly by Node RED servers in the cloud.

▪ Cheaper and low requirement micro controllers can be used since processing requirements are reduced.

▪ Longer lasting systems.

▪ Dynamic Sign updating.

▪ School/Hospital Zone alerts

•DISADAVNTAGES

▪ The size of the display determines the requirement of the micro controller.

▪ Dependent on OpenWeatherMap API and hence the speed reduction is same for a large area in the scale of Cities.

CHAPTER 11

11. CONCLUSION :

Our project is capable of serving as a replacement for
static signs for a comparatively lower cost and can be implemented in the very near future. This will help reduce a lot of accidents and maintain a more peaceful traffic atmosphere in the country.

GitHub link:
https://github.com/IBM-EPBL/IBM-Project-9461-1659009035.