Revista de Investigación Multidisciplinaria

CTSCAFE PARA CIUDADANOS.....

http://www.ctscafe.pe

ISSN 2521-8093

Volumen VI- Nº 17 Julio 2022

http://www.ctscafe.pe

Lima - Perú

REVISTA DE INVESTIGACIÓN MULTIDISCIPLINARIA

CTS CAFE Para ciudadanos…

http://www.ctscafe.pe

Volumen VI- N° 17 Julio 2022

ISSN 2521-8093

Análisis de confiabilidad del procedimiento de cálculo de perfiles de velocidad en camiones mineros

Anthony Enrique Arrasco Napa
Universidad Nacional de Ingeniería
Correo Electrónico: anthony.arrasco.n@uni.pe

Luis Fernando Flores Ponte
Universidad Nacional de Ingeniería
Correo Electrónico: luis.flores.p@uni.pe

Emanuel Piero Poma Huamán
Universidad Nacional de Ingeniería
Correo Electrónico: epomah@uni.pe

Resumen: El objetivo de este artículo es realizar un análisis de sensibilidad al trabajar con distintas muestras de una base de datos proporcionada por el sistema de despacho de una mina y determinar los casos en los cuales se obtiene un grado de confiabilidad del 95%. El procedimiento de cálculo del perfil de velocidades de la flota de camiones de acarreo en función de la gradiente, desarrollado en el lenguaje de programación Python, reportará distintos resultados para intervalos de cada 10% del total de datos, hasta completar el 100% de la data. Este análisis se realiza tanto para un mes de temporada de lluvias (e.g. enero) como para un mes de temporada seca (e.g. mayo).

Palabras claves: Sensibilizar/ Confiabilidad/ Perfil de velocidades/ Estado de camión/ Programación.

Abstract: The objective of this article is to carry out a sensitivity analysis by working with different samples of a database provided by the dispatch system of a mine and to determine the cases in which a degree of reliability of 95% is obtained. The procedure for calculating the speed profile of the haul truck fleet as a function of the gradient, developed in the Python programming language, will report different results for intervals of each 10% of the total data, until completing 100% of the data. This analysis is performed for both a rainy season month (e.g. January) and a dry season month (e.g. May).

Keywords: Sensitize / Reliability / Speed profile / Truck status / Programming.

Résumé : L'objectif de cet article est de réaliser une analyse de sensibilité en travaillant avec différents échantillons d'une base de données fournie par le dispatching d'une mine et de déterminer les cas dans lesquels un degré de fiabilité de 95% est obtenu. La procédure de calcul du profil de vitesse de la flotte de camions de transport en fonction du gradient, développée dans le langage de programmation Python, rapportera des résultats différents pour des intervalles de chaque 10 % des données totales, jusqu'à compléter 100 % des données. Cette analyse est effectuée à la fois pour un mois de saison des pluies (par exemple janvier) et un mois de saison sèche (par exemple mai).

Mots-clés: Sensibiliser / Fiabilité / Profil de vitesse / Etat du camion / Programmation.

1. Introducción

La operación de acarreo es uno de los principales factores críticos en el ciclo del proceso productivo de una mina debido a los costos asociados que involucra, así como la influencia que posee frente a la productividad global de la operación minera. Por tales razones, es imprescindible conocer cómo se comporta la flota de camiones mineros a lo largo de las rutas de acarreo a través de distintas pendientes, diferenciando el estado de estos; es decir, si el camión está cargado o descargado; y tomando en consideración la temporada sobre la cual se realiza el estudio, si esta es en temporada seca o bajo una temporada de lluvias.

El presente trabajo muestra el cálculo de velocidades respecto a gradientes, a partir de la data cruda entregada por el sistema de despacho de camiones, tomando en consideración variables, como coordenadas UTM, pendiente del terreno y el tonelaje para discriminar el estado de los camiones en cargados y vacíos. Esta investigación muestra el perfil de velocidades de los camiones en función de la gradiente de -12% a 12% para 2 meses de distintas temporadas, una temporada seca como enero y otra de lluvias como mayo. A partir de los resultados obtenidos para estos meses, se procedió a sensibilizar el cálculo de velocidades para distintos rangos de valores de la base de datos, esto a fin de estimar las velocidades de los camiones mineros a una pendiente de 0%. Se escogieron muestras cada 10%, 20%, y así sucesivamente hasta realizar el cálculo con el 100 % de la base de datos del sistema de despacho, mediante un algoritmo desarrollado en el lenguaje de programación Python.

2. Material y métodos

El algoritmo presentado en el informe fue desarrollado en el lenguaje de programación Python, por la facilidad que representa trabajar en este lenguaje. Los entornos de ejecución fueron JupyterLab y Google Colaboratory. Las principales librerías usadas fueron Pandas, Matplotlib.Pyplot, Math y Numpy. Las características del equipo sobre el cual se ejecutó el algoritmo son las siguientes: procesador Intel Core I7-10700F @ 2.90 GHz y una memoria RAM de 16 GB.

El algoritmo diseñado en el lenguaje de programación Python para obtener las velocidades y los resultados deseados resultó muy eficiente para trabajar con grandes cantidades de datos; sin embargo, cabe mencionar que la capacidad de memoria RAM es una limitante para poder analizar el 100% de la data. A continuación, se detallará la metodología desarrollada en la presente investigación.

PASO 1:

El primer paso fue cargar la data a Python, previamente los archivos en formato .csv fueron guardados en una carpeta local de nuestro ordenador con el nombre de perfil de velocidades, en este ejemplo escogeremos el mes de mayo (archivo de nombre 2019-05.csv. El código usado fue el siguiente:

Figura Nº 1: Importación de archivo

```
import pandas as pd
archivo=pd.read_csv("C:/Users/Anthony/Desktop/perfil_velocidades/2019-05.csv",header=0)
```

Fuente: Elaboración propia

Al imprimir en pantalla la data cargada, se obtuvo un DataFrame de Pandas con las columnas (id, id_equipo, id_flota, nombre_equipo, id_dcamión, xcoorint, ycoorint, xcoor, ycoor, velocidad, tonelaje, incl_roll, incl_pitch, tramosids, tiem_creac, id_data_ca)

Figura Nº 2: Cinco primeros registros de la base de datos

```
archivo.head()
```

	id	id_equipo	id_flota	nombre_equipo	id_dcamion	xcoorint	ycoorint	xcoor	
0	39	39	22	CM12	345051	20086401.0	20086401.0	{20008792,20008792,20008793,20008793,20008793,...	{839877841,839877820,839877
1	38	38	22	CM11	345050	20134201.0	20134201.0	{20115084,20115084,20115002,20115003,20115003,...	{839862687,839862687,839862
2	35	35	22	CM08	345049	20151663.0	20151663.0	{20115940,20115940,20115940,20115940,20115940,...	{839866774,839866774,839866
3	31	31	22	CM04	345048	20181064.0	20181064.0	{20019362,20017136,20015814,20013915,20013337,...	{839875150,839875397,839875
4	42	42	22	CM15	345047	20046605.0	20046605.0	{20190857,20190857,20190857,20190857,20190857,...	{839941176,839941187,839941

Fuente: Elaboración propia

PASO 2:

De la data cargada del paso 1, solo seleccionamos las columnas con las cuales trabajaremos, para el caso son xcoor, ycoor, incl_pitch y tonelaje, mediante el siguiente código:

Figura Nº 3: Selección de las columnas a usar

```
xcoor=pd.DataFrame(archivo['xcoor'],columns=['xcoor'])
ycoor=pd.DataFrame(archivo['ycoor'],columns=['ycoor'])
incl_pitch=pd.DataFrame(archivo['incl_pitch'],columns=['incl_pitch'])
tonelaje=pd.DataFrame(archivo['tonelaje'],columns=['tonelaje'])
```

Fuente: Elaboración propia

PASO 3:

Al seleccionar las columnas indicadas del paso anterior, se observó un problema al presentar llaves iniciales y finales en cada una de estas; para ejemplificar esto, al imprimir xcoor en pantalla se obtiene:

Figura Nº 4: Cinco primeros registros de la columna xcoor

	xcoor
0	{20008792,20008792,20008793,20008793,20008793,...
1	{20115084,20115084,20115002,20115003,20115003,...
2	{20115940,20115940,20115940,20115940,20115940,...
3	{20019362,20017136,20015814,20013915,20013337,...
4	{20190857,20190857,20190857,20190857,20190857,...

Fuente: Elaboración propia

Por ende, se procedió a eliminar las llaves de las cuatro columnas (xcoor, ycoor, incl_pitch y tonelaje) con el siguiente código:

Figura Nº 5: Código para limpieza de los registros

```
for i in range(0,xcoor.shape[0]):
    elemento=xcoor.values[i][0]
    elemento=elemento.replace('{','')
    elemento=elemento.replace('}','')
    #elemento=elemento.replace("'","")
    xcoor.values[i][0]=elemento
for i in range(0,ycoor.shape[0]):
    elemento=ycoor.values[i][0]
    elemento=elemento.replace('{','')
    elemento=elemento.replace('}','')
    #elemento=elemento.replace("'","")
    ycoor.values[i][0]=elemento
```

```
for i in range(0,tonelaje.shape[0]):
    elemento=tonelaje.values[i][0]
    elemento=elemento.replace('{','')
    elemento=elemento.replace('}','')
    #elemento=elemento.replace("'","")
    tonelaje.values[i][0]=elemento
for i in range(0,incl_pitch.shape[0]):
    elemento=incl_pitch.values[i][0]
    elemento=elemento.replace('{','')
    elemento=elemento.replace('}','')
    #elemento=elemento.replace("'","")
    incl_pitch.values[i][0]=elemento
elemento=0
```

Fuente: Elaboración propia

PASO 4:

Ya eliminadas las llaves con el script anterior, se realizó un procedimiento que permitió variar el porcentaje de la data a analizar. Tal y como se enunció en la introducción, se fue aumentando el valor de la muestra progresivamente mediante un incremento de 10% de la base de datos total.

Figura Nº 6: Selección del porcentaje de la data a utilizar

```
1 import math
2 porc=0.5
3 total=xcoor.shape[0]*porc
4 total=math.ceil(total)
5 print(total)
```

En este ejemplo se tomó el 50% de la data

```
7316
```

```
1 xcoor=xcoor.loc[0:total,:]
2 ycoor=ycoor.loc[0:total,:]
3 tonelaje=tonelaje.loc[0:total,:]
4 incl_pitch=incl_pitch.loc[0:total,:]
```

Fuente: Elaboración propia

PASO 5:

En este paso se muestra el script usado para poder expandir los datos de los DataFrame: xcoor, ycoor, incl_pitch en columnas usando como criterio de separación, las comas (,).

Figura Nº 7: Expansión si de las variables en filas

```
1 xcoor=xcoor['xcoor'].str.split(",",expand=True)
2 ycoor=ycoor['ycoor'].str.split(",",expand=True)
3 tonelaje=tonelaje['tonelaje'].str.split(",",expand=True)
4 incl_pitch=incl_pitch['incl_pitch'].str.split(",",expand=True)
```

Fuente: Elaboración propia

Se obtendrá cada columna previa ya expandida, por ejemplo xcoor será de la forma:

Figura Nº 8: Cinco primeros registros de la columna xcoor expandida

```
1 xcoor
```

	0	1	2	3	4	5	6	7	8	9	...	1790	1791	1792	1793	
0	20117344	20116178	20115675	20115249	20115327	20115648	20116383	20116629	20117041	20117123	...	None	None	None	None	
1	20166452	20166452	20166451	20166451	20166451	20166451	20166451	20166369	20166368	20166368	...	20179091	20179091	20179009	20179009	2
2	20124678	20124761	20124761	20124761	20124761	20124761	20124761	20124761	20124761	20124760	...	20066883	20068688	20070329	20072052	2
3	20185110	20185110	20185110	20185110	20185110	20185110	20185110	20185110	20185110	20185110	...	20005781	20003392	19999602	19998284	1
4	20137100	20137839	20139978	20141542	20143270	20146641	20146641	20147956	20152313	20155434	...	19925470	19923336	19921203	19919155	1

Fuente: Elaboración propia

PASO 6:

Luego de haber expandido las columnas, se procedió a convertir los valores string a valores float con el siguiente script, para poder realizar los cálculos correspondientes.

Figura Nº 9: Conversión de columnas y tipo de datos de las variables

```
#convertimos a dataframe los xcoor en una sola columna:
xcoor=pd.DataFrame(xcoor.values.reshape(xcoor.values.shape[0]*xcoor.values.shape[1],1),columns=['xcoor'])
ycoor=pd.DataFrame(ycoor.values.reshape(ycoor.values.shape[0]*ycoor.values.shape[1],1),columns=['ycoor'])
tonelaje=pd.DataFrame(tonelaje.values.reshape(tonelaje.values.shape[0]*tonelaje.values.shape[1],1),columns=['tonelaje'
incl_pitch=pd.DataFrame(incl_pitch.values.reshape(incl_pitch.values.shape[0]*incl_pitch.values.shape[1],1),
                        columns=['incl_pitch'])
#convertimos a valores float los valores con los que haremos cálculos
xcoor=pd.to_numeric(xcoor['xcoor'],errors='coerce')
ycoor=pd.to_numeric(ycoor['ycoor'],errors='coerce')
incl_pitch=pd.to_numeric(incl_pitch['incl_pitch'],errors='coerce')
#volvemos a convertir en DataFrame xcoor,ycoor,incl_pitch y tonelaje ya previamente está convertido en DataFrame
xcoor=pd.DataFrame(xcoor,columns=['xcoor'])
ycoor=pd.DataFrame(ycoor,columns=['ycoor'])
incl_pitch=pd.DataFrame(incl_pitch,columns=['incl_pitch'])
```

Fuente: Elaboración propia

PASO 7:

En este paso se procedió al cambio de unidades correctas para el cálculo, cabe resaltar que la data proporcionada tenía las coordenadas X y Y multiplicadas por 100, el ángulo de inclinación multiplicado por 100 y el tonelaje multiplicado por 10.

- Se dividió entre 100 a xcoor, ycoor, incl_pitch, el tonelaje se obvio debido a que no involucra cálculo alguno, sino una discriminación cuando es cero o no.

Figura Nº 10: Cambio de unidades de las columnas

```
xcoor['xcoor']=xcoor['xcoor']/100
ycoor['ycoor']=ycoor['ycoor']/100
incl_pitch['incl_pitch']=incl_pitch['incl_pitch']/100
```

Fuente: Elaboración propia

- Para controlar valores nulos, se optó por reemplazarlos con un valor aleatoriamente elegido (-80 en este caso), esto no traerá ningún problema, ya que al filtrar los valores obtenidos estos serán ignorados porque arrojarán valores no válidos y fuera del rango del análisis (valores outliers).

Figura Nº 11: Reemplazo de valores faltantes por valores outliers

```
xcoor=xcoor.fillna(-80)
ycoor=ycoor.fillna(-80)
incl_pitch=incl_pitch.fillna(-80)
```

Fuente: Elaboración propia

- Para poder agrupar las velocidades por gradientes, se necesita que el incl_pitch expresado en grados sexagesimales (°) se transforme a porcentaje (%), mediante el siguiente script se obtiene lo requerido en un DataFrame.

Figura Nº 12: Conversión del ángulo de inclinación de grados sexagesimales a porcentaje (%)

```
import math
import numpy as np
valores=incl_pitch['incl_pitch'].values*(math.pi/180) # convirtiendo a radianes
valores=np.tan(valores)*100 # convirtiendo a %
```

```
data2=pd.DataFrame(valores,columns=['incl_pitch_%'])
data2
```

Fuente: Elaboración propia

- Obteniéndose los siguientes resultados:

	incl_pitch_%
0	-5.433311
1	-4.943302
2	-3.771698
3	-6.501763
4	-5.800987
...	...
13170595	18.461703
13170596	17.326926
13170597	17.578715
13170598	16.985553
13170599	11.146111

13170600 rows × 1 columns

Fuente: Elaboración propia

PASO 8:

Para poder hacer el cálculo de forma eficiente y ahorrar costo computacional, se convirtieron las columnas del DataFrame xcoor, ycoor, incl_pitch y tonelaje a matrices de la librería Numpy de Python, asignando el número de filas y columnas iniciales obtenidas al expandir los DataFrame en el paso 5, mediante el siguiente script:

Figura Nº 13: Remodelamiento de matrices

```
#convertimos los dataframes separados a matrices
xcoor_matriz=xcoor['xcoor'].to_numpy()
ycoor_matriz=ycoor['ycoor'].to_numpy()
tonelaje_matriz=tonelaje['tonelaje'].to_numpy()
incl_pitch_matriz=incl_pitch['incl_pitch'].to_numpy()
incl_pitch_por_matriz=data2['incl_pitch_%'].to_numpy()
#remodelamos las matrices
xcoor_matriz=xcoor_matriz.reshape(xcoor.shape[0]//1800,1800)
ycoor_matriz=ycoor_matriz.reshape(ycoor.shape[0]//1800,1800)
tonelaje_matriz=tonelaje_matriz.reshape(tonelaje_matriz.shape[0]//1800,1800)
incl_pitch_matriz=incl_pitch_matriz.reshape(incl_pitch.shape[0]//1800,1800)
incl_pitch_por_matriz=incl_pitch_por_matriz.reshape(incl_pitch.shape[0]//1800,1800)
```

Fuente: Elaboración propia

PASO 9:

En este paso, se procedió a calcular las velocidades y agruparlas en base a los valores de la columna incl_pitch_% (gradientes desde -12% a 12%), mediante el siguiente script:

Figura Nº 14: Agrupamiento según gradiente y estado del camión (cargado o vacío)


```
long=(xcoor_matriz.shape[1]-1)*xcoor_matriz.shape[0]
velocidades=np.zeros(long)
velocidades=velocidades.reshape(1799,long//1799)
gradientes=np.zeros(long)
gradientes=gradientes.reshape(1799,long//1799)
estado=np.zeros(long)
estado=estado.reshape(1799,long//1799)
m=0
n=xcoor_matriz.shape[1]-1 #1799
k=xcoor_matriz.shape[0]   #10000

for i in range(m,n):
    d=((xcoor_matriz.T[i]-xcoor_matriz.T[i+1])**2+(ycoor_matriz.T[i]-ycoor_matriz.T[i+1])**2)**(0.5)
    v=(d/np.cos(incl_pitch_matriz.T[i]*math.pi/180))*0.5*3.6
    velocidades[i]=v
    ton_bool=tonelaje_matriz.T[i]=="0"
    estado[i]=ton_bool
    ############################
    incl=np.zeros(k)
    x=incl_pitch_por_matriz.T[i]
    incl_bool=np.ceil(x)-x>=x-np.floor(x)
    for j in range(0,k):
        if incl_bool[j]==True:
            incl[j]=np.floor(x[j])
        else:
            incl[j]=np.ceil(x[j])
    gradientes[i]=incl
#remodelamos las matrices
velocidades=velocidades.T.reshape(long,1)
gradientes=gradientes.T.reshape(long,1)
estado=estado.T.reshape(long,1)
df_v=pd.DataFrame(velocidades,columns=['velocidad'])
df_g=pd.DataFrame(gradientes,columns=['gradiente'])
df_e=pd.DataFrame(estado,columns=['estado'])
data=pd.concat([df_v,df_g,df_e],axis=1)
data
```

Fuente: Elaboración propia

- Obteniéndose los siguientes resultados:

Figura Nº 15: Resultados agrupamiento por gradiente y estado

	velocidad	gradiente	estado
0	28.309489	-5.0	1.0
1	16.349760	-5.0	1.0
2	22.903045	-4.0	1.0
3	7.404222	-7.0	1.0
4	13.200380	-6.0	1.0
...	...	...	...
13163278	12.200064	19.0	0.0
13163279	36.449613	18.0	0.0
13163280	0.000000	17.0	0.0
13163281	22.781927	18.0	0.0
13163282	25.611383	17.0	0.0

13163283 rows × 3 columns

Fuente: Elaboración propia

PASO 10:

El DataFrame que se obtuvo en el paso anterior ya está listo para filtrarse y obtener el perfil de velocidades:

- Separación del DataFrame en 2 categorías, empty y loaded, a través del siguiente script:

Figura Nº 16: Separación según estado del camión.

```
data_vacio=data.loc[(data['estado']==True)]
data_cargado=data.loc[(data['estado']==False)]
```

Fuente: Elaboración propia

- El siguiente script permite crear una lista previa de las gradientes (desde -12% a 12%).

Figura Nº 17: Creación de las listas para los valores de gradientes

```
lista_gradientes=[]
for i in range(-12,13):
    lista_gradientes.append(i)
```

Fuente: Elaboración propia

- Por último, se realiza el cálculo de las medias, medianas y modas de cada gradiente y lo almacenamos en una hoja de Excel en la dirección local que se elija, con el siguiente script, tanto para empty como para loaded.

Figura Nº 18: Cálculo de la media, mediana y moda para las velocidades

Para obtener resultados de camiones vacíos, solo modificamos el script con el nombre data_vacio creado y filtrado previamente en la primera instrucción de este paso.

```
1
2  v_media=[]
3  v_moda=[]
4  v_mediana=[]
5  for i in lista_gradientes:
6      df_g=data_cargado.loc[(data_cargado['gradiente'])==i]
7      df_g=df_g.loc[(df_g['velocidad'])>=10]
8      df_g=df_g.loc[(df_g['velocidad'])<=65]
9      v_media_g=round(df_g['velocidad'].mean(),2)
10     v_mediana_g=round(df_g['velocidad'].median(),2)
11     v_moda_g=round(df_g['velocidad'].mode()[0],2)
12     v_media.append(v_media_g)
13     v_mediana.append(v_mediana_g)
14     v_moda.append(v_moda_g)
15
16 dicc = {
17     "gradientes": lista_gradientes,
18     "v_media": v_media,
19     "v_moda":v_moda,
20     "v_mediana":v_mediana
21
22 }
23
24 df = pd.DataFrame(dicc)
```

Fuente: Elaboración propia

PASO 11:

Se elige la dirección donde se guardará el resultado obtenido, como ya se mencionó estos resultados representan los cálculos para mayo, trabajando con un 50% de la data y considerando los camiones cargados, el mismo procedimiento se repitió para los demás casos.

Este script almacenará los resultados en una hoja de Excel en la dirección que elijamos de la forma:

Figura Nº 19: Dirección de salida del archivo a generar con el reporte de velocidades

```
df.to_excel('C:/Users/Anthony/Desktop/perfil_velocidades/pv_05_c_porcentajex.xlsx')
```

	A	B	C	D	E
1		gradientes	v_media	v_moda	v_mediana
2	0	-12	22,42	22,2	22,58
3	1	-11	22,92	22,2	22,64
4	2	-10	23,42	22,17	22,77
5	3	-9	24,59	11,54	23
6	4	-8	26,11	11,55	23,64
7	5	-7	26,62	11,34	23,76
8	6	-6	26,5	22,29	23,55
9	7	-5	26,47	22,08	23,52
10	8	-4	26,32	10,5	23,22
11	9	-3	26,18	11,89	22,74
12	10	-2	26,48	11,87	23
13	11	-1	26,42	11,85	23,1
14	12	0	24,23	22,05	22,37
15	13	1	26,56	11,86	23,75
16	14	2	25,84	12,52	23,42
17	15	3	25,06	11,61	22,88
18	16	4	23,91	12,05	22,12
19	17	5	22,6	11,86	20,86
20	18	6	21,59	12,06	19,3
21	19	7	20,41	11,9	17,47
22	20	8	19,6	10,53	16,42
23	21	9	20,12	10,53	17,16
24	22	10	20,54	11,95	17,69
25	23	11	19,79	10,56	16,24
26	24	12	19,58	11,31	15,37

Nombre del archivo Excel :

pv ➔ perfil de velocidades

05 ➔ número de mes (mayo en este caso)

c ➔ Cargado , v si fueran camiones vacíos

Porcentajes ➔ Representa el porcentaje de la data con la que trabaja en cada ejecución (para identificación)

Fuente: Elaboración propia

3. Resultados

El análisis para la temporada de lluvias, en este caso representada por el mes de enero, se muestra a continuación en las siguientes tablas.

Tabla Nº 1: Estimación de velocidades para **camiones vacíos** en el mes de enero para distintas muestras

	SPEED EMPTY TRUCKS (km/h) - JANUARY									
GRADIENT	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
-12%	25.0	25.5	26.0	26.4	27.0	27.2	27.1	27.4	27.9	28.3
-11%	25.4	26.0	26.4	26.9	27.3	27.6	27.5	28.0	28.3	28.5
-10%	25.3	26.5	26.9	27.2	27.6	27.9	27.9	28.4	28.6	28.8
-9%	25.4	26.7	27.2	27.5	27.8	28.1	28.2	28.8	29.1	29.4
-8%	25.7	26.9	27.6	28.0	28.2	28.4	28.5	29.1	29.4	29.7
-7%	25.0	26.5	27.4	27.9	28.2	28.4	28.4	28.8	29.1	29.5
-6%	24.1	25.5	26.7	27.2	27.5	27.9	27.8	28.3	28.6	29.0
-5%	23.7	25.4	26.5	27.2	27.4	27.7	27.6	28.2	28.4	28.7
-4%	23.5	25.2	26.3	27.1	27.4	27.7	27.6	28.2	28.4	28.7
-3%	23.8	25.3	26.5	27.1	27.4	27.8	27.7	28.2	28.5	28.7
-2%	24.6	25.8	26.8	27.3	27.6	28.0	27.9	28.3	28.5	28.7
-1%	25.5	26.4	27.1	27.7	28.1	28.2	28.2	28.5	28.7	28.9
0%	26.4	27.2	28.0	28.2	28.5	28.6	28.7	29.0	29.2	29.3
1%	26.1	27.0	27.8	28.1	28.3	28.5	28.4	28.7	28.9	29.0
2%	26.3	27.2	27.7	27.9	28.1	28.2	28.2	28.4	28.5	28.6
3%	26.2	27.0	27.4	27.8	28.0	28.2	28.2	28.5	28.7	28.8
4%	26.7	27.5	28.0	28.4	28.7	28.9	28.9	29.3	29.5	29.7
5%	27.0	27.7	28.2	28.6	28.9	29.1	29.2	29.5	29.8	29.8
6%	27.1	27.6	28.0	28.3	28.6	28.8	29.0	29.2	29.5	29.6
7%	26.7	27.2	27.5	27.8	28.1	28.3	28.3	28.6	28.8	29.0
8%	26.2	26.7	26.9	27.2	27.5	27.8	27.8	28.2	28.4	28.6
9%	25.5	26.0	26.4	26.8	27.1	27.4	27.4	27.8	28.3	28.4
10%	25.5	26.0	26.7	27.1	27.4	27.6	27.7	28.2	28.6	28.8
11%	26.9	27.5	28.3	28.8	29.1	29.2	29.3	29.8	30.0	30.2
12%	28.4	29.0	29.6	29.8	29.9	30.0	30.3	30.5	30.7	30.8

Fuente: Elaboración propia

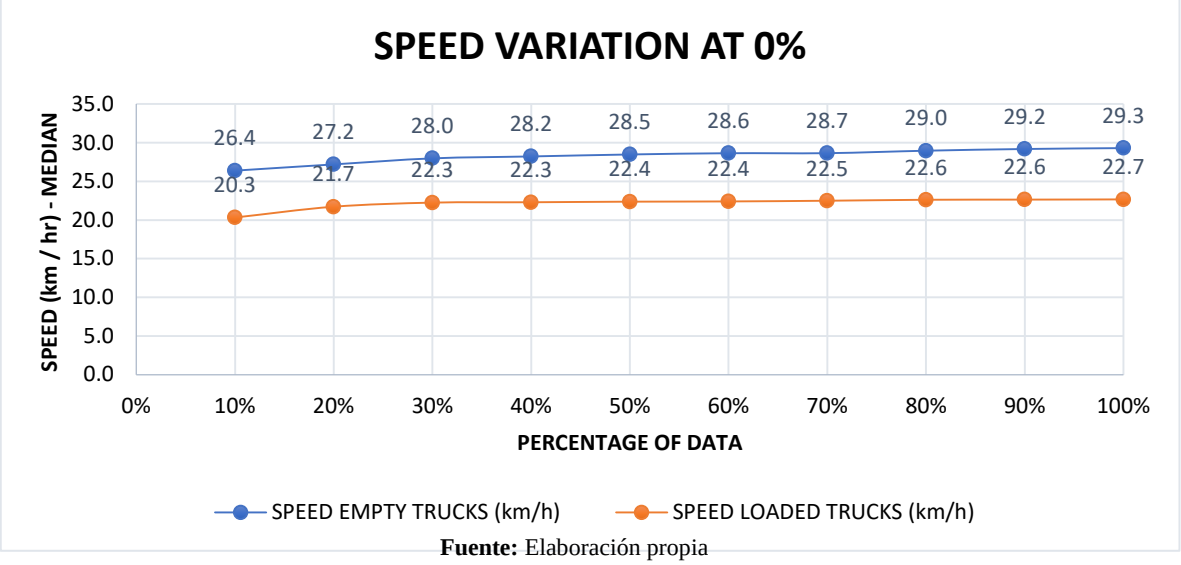
Tabla Nº 2: Estimación de velocidades para **camiones cargados** en el mes de enero para distintas muestras

	SPEED LOADED TRUCKS (km/h) - JANUARY									
GRADIENT	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
-12%	22.4	22.6	22.8	22.8	22.8	22.8	22.9	22.9	22.9	22.8
-11%	22.4	22.6	22.9	23.0	23.0	23.0	23.0	23.0	23.0	23.0
-10%	22.6	22.8	23.1	23.2	23.2	23.2	23.2	23.2	23.2	23.3
-9%	22.8	23.0	23.2	23.3	23.3	23.4	23.4	23.5	23.5	23.5
-8%	22.9	23.1	23.3	23.3	23.5	23.5	23.5	23.5	23.6	23.6
-7%	22.9	23.1	23.2	23.3	23.4	23.5	23.5	23.6	23.6	23.6
-6%	22.9	23.0	23.1	23.3	23.5	23.5	23.6	23.6	23.7	23.6
-5%	22.7	22.9	23.1	23.3	23.5	23.6	23.7	23.7	23.8	23.8
-4%	21.9	22.3	22.6	22.9	23.2	23.4	23.5	23.6	23.7	23.8
-3%	19.9	20.8	21.6	22.1	22.4	22.5	22.7	22.9	23.0	23.1
-2%	18.8	19.7	20.8	21.1	21.5	21.6	21.8	22.2	22.3	22.3
-1%	19.0	19.8	20.8	21.0	21.3	21.4	21.7	22.1	22.2	22.2
0%	20.3	21.7	22.3	22.3	22.4	22.4	22.5	22.6	22.6	22.7
1%	19.9	21.1	22.1	22.3	22.4	22.5	22.5	22.9	22.9	23.1
2%	20.5	21.7	22.4	22.6	22.5	22.6	22.7	23.0	22.9	23.1
3%	20.0	21.9	22.4	22.4	22.2	22.2	22.3	22.5	22.4	22.5
4%	18.9	21.1	22.2	22.2	21.8	21.9	21.9	22.3	22.3	22.3
5%	17.9	20.8	21.8	21.8	21.7	21.7	21.9	22.3	22.3	22.4
6%	16.8	19.5	21.2	21.1	21.0	21.2	21.4	22.0	22.1	22.3
7%	15.8	17.9	19.6	19.3	19.0	19.3	19.7	20.9	21.0	21.2
8%	15.1	16.6	17.6	17.5	17.4	17.7	17.9	18.5	18.5	19.0
9%	14.1	16.4	16.7	16.7	16.7	16.9	17.2	17.7	17.7	17.9
10%	14.0	16.5	17.0	17.4	17.6	17.9	18.0	18.3	18.2	18.4
11%	14.9	16.5	16.9	17.8	18.0	18.2	18.6	18.8	18.7	18.9
12%	15.8	16.8	17.4	18.3	18.5	19.4	19.6	19.8	19.8	19.8

Fuente: Elaboración propia

Para este primer mes, se obtuvo la siguiente relación para distintas proporciones de muestra, donde se observa que, tanto para los camiones cargados como vacíos, a medida que se toma una menor cantidad de data existe una subestimación de la velocidad real. La gráfica muestra el comportamiento de las velocidades a una pendiente de 0%.

Figura Nº 20: Estimación de velocidad a una gradiente de 0% variando intervalos cada 10%



Fuente: Elaboración propia

El análisis para la temporada seca, en este caso representada por el mes de mayo, se muestra a continuación en las siguientes tablas.

Tabla Nº 3: Estimación de velocidades para **camiones vacíos** en el mes de mayo para distintas muestras

	SPEED EMPTY TRUCKS (km/h) - MAY									
GRADIENT	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
-12%	31.5	31.5	31.8	31.6	31.2	31.0	30.8	30.8	30.9	31.1
-11%	31.5	31.3	31.3	31.3	30.9	30.6	30.4	30.3	30.3	30.3
-10%	32.4	32.4	32.3	32.2	32.0	31.8	31.5	31.4	31.3	31.3
-9%	33.6	33.3	33.3	33.2	33.1	32.8	32.5	32.4	32.2	32.2
-8%	34.4	34.4	34.3	33.9	33.8	33.4	33.0	32.8	32.7	32.7
-7%	34.0	34.1	34.0	33.4	33.3	32.9	32.4	32.2	32.2	32.3
-6%	32.5	32.8	32.7	32.4	32.2	31.6	31.3	31.2	31.2	31.3
-5%	32.0	32.3	32.2	31.6	31.4	31.0	30.6	30.6	30.6	30.8
-4%	31.6	31.9	31.6	31.2	31.0	30.4	30.1	30.0	30.0	30.2
-3%	31.2	31.2	30.8	30.4	30.2	29.8	29.6	29.5	29.6	29.7
-2%	31.0	31.0	30.8	30.4	30.2	29.9	29.7	29.5	29.5	29.6
-1%	32.2	32.1	31.8	31.6	31.3	31.0	30.6	30.3	30.2	30.2
0%	31.6	31.7	31.8	31.5	31.3	31.1	30.9	30.8	30.8	30.9
1%	33.4	33.4	33.2	33.1	32.8	32.3	31.6	31.3	31.3	31.3
2%	33.1	33.7	33.5	33.6	33.3	32.7	32.1	31.8	31.7	31.8
3%	33.6	33.9	33.7	33.8	33.6	33.0	32.6	32.4	32.3	32.3
4%	32.6	32.9	33.0	33.0	32.9	32.5	32.2	32.0	32.0	32.0
5%	31.6	31.7	31.9	31.9	31.8	31.6	31.3	31.2	31.2	31.2
6%	31.3	31.3	31.5	31.5	31.4	31.2	31.0	30.8	30.7	30.8
7%	31.6	31.4	31.6	31.6	31.5	31.3	31.1	30.8	30.8	30.8
8%	32.2	31.7	31.7	31.4	31.2	31.0	30.8	30.5	30.5	30.5
9%	31.9	31.4	31.2	30.8	30.3	30.1	29.9	29.8	29.8	29.8
10%	30.8	30.6	30.4	29.9	29.5	29.4	29.2	29.2	29.2	29.2
11%	30.3	30.9	30.8	30.4	30.1	29.9	29.9	30.0	30.0	30.2
12%	30.6	31.3	31.4	31.4	31.4	31.0	30.8	30.9	31.0	31.1

Fuente: Elaboración propia

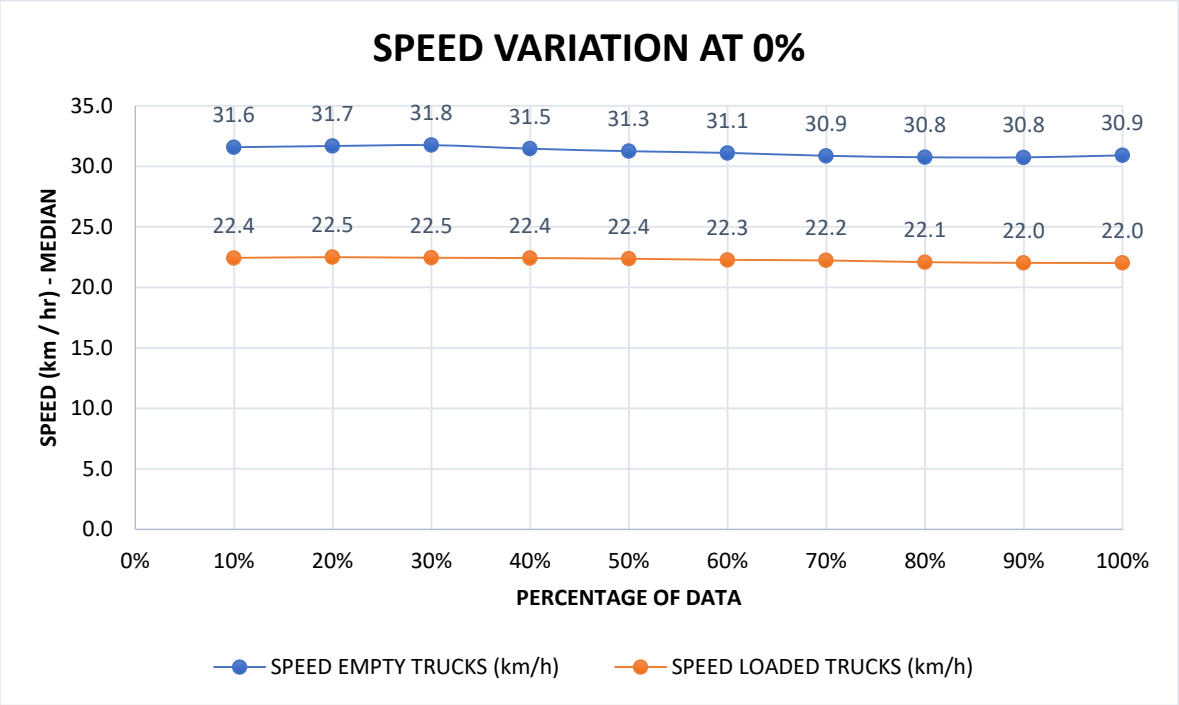
Tabla N° 4: Estimación de velocidades para **camiones cargados** en el mes de mayo para distintas muestras

	SPEED LOADED TRUCKS (km/h)									
GRADIENT	**10%**	**20%**	**30%**	**40%**	**50%**	**60%**	**70%**	**80%**	**90%**	**100%**
-12%	22.2	22.1	22.3	22.5	22.6	22.5	22.4	22.4	22.4	22.3
-11%	22.6	22.4	22.4	22.6	22.6	22.6	22.5	22.4	22.4	22.3
-10%	22.8	22.6	22.6	22.8	22.8	22.6	22.6	22.5	22.4	22.4
-9%	23.0	22.9	22.9	23.0	23.0	22.8	22.7	22.7	22.6	22.5
-8%	23.5	23.6	23.5	23.7	23.6	23.5	23.3	23.3	23.3	23.2
-7%	23.5	23.6	23.7	23.8	23.8	23.7	23.5	23.5	23.5	23.5
-6%	23.3	23.5	23.5	23.5	23.6	23.6	23.5	23.5	23.4	23.5
-5%	23.4	23.6	23.4	23.5	23.5	23.5	23.4	23.4	23.4	23.5
-4%	23.1	23.4	23.1	23.1	23.2	23.2	23.1	23.1	23.1	23.1
-3%	22.8	23.1	22.8	22.8	22.7	22.7	22.6	22.3	22.3	22.3
-2%	24.0	24.0	23.5	23.2	23.0	22.8	22.6	22.2	21.9	21.9
-1%	24.0	24.2	23.8	23.2	23.1	23.0	22.8	22.4	22.3	22.3
0%	22.4	22.5	22.5	22.4	22.4	22.3	22.2	22.1	22.0	22.0
1%	24.1	24.6	24.4	23.8	23.8	23.6	23.4	23.2	23.1	23.2
2%	23.4	23.8	23.8	23.4	23.4	23.2	23.2	23.0	22.9	23.0
3%	22.7	23.3	23.2	22.9	22.9	22.7	22.7	22.7	22.7	22.8
4%	21.7	22.4	22.3	22.2	22.1	21.8	21.9	22.2	22.3	22.3
5%	20.7	21.8	21.4	20.9	20.9	20.5	20.8	21.1	21.3	21.4
6%	19.4	21.0	19.7	19.3	19.3	18.9	19.0	19.3	19.5	19.7
7%	17.8	18.1	17.8	17.4	17.5	17.2	17.2	17.3	17.5	17.6
8%	16.7	17.0	16.5	16.4	16.4	16.4	16.4	16.4	16.4	16.4
9%	17.9	18.0	17.2	16.8	17.2	17.0	17.0	17.0	16.9	16.9
10%	19.2	19.4	17.9	17.3	17.7	17.5	17.4	17.4	17.2	17.1
11%	16.5	17.1	16.3	16.0	16.2	16.1	16.1	16.2	16.4	16.4
12%	14.9	15.6	15.0	15.2	15.4	15.4	15.2	15.2	15.5	15.7

Fuente: Elaboración propia

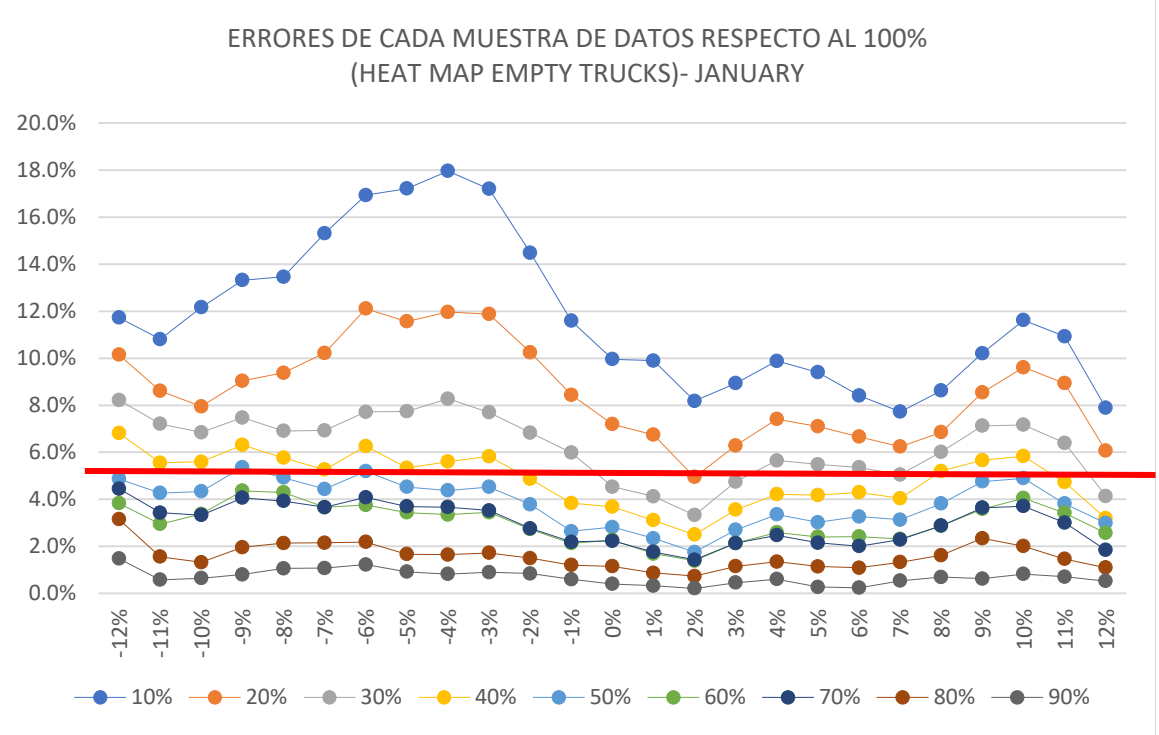
Para este segundo mes, se obtuvo la siguiente relación para distintas proporciones de muestra, donde se observa que tanto para los camiones cargados como vacíos, a medida que se toma una menor cantidad de data existe una sobreestimación de la velocidad real. La gráfica muestra el comportamiento de las velocidades a una pendiente de 0%.

Figura N° 21: Estimación de velocidad a una gradiente de 0% variando intervalos cada 10%



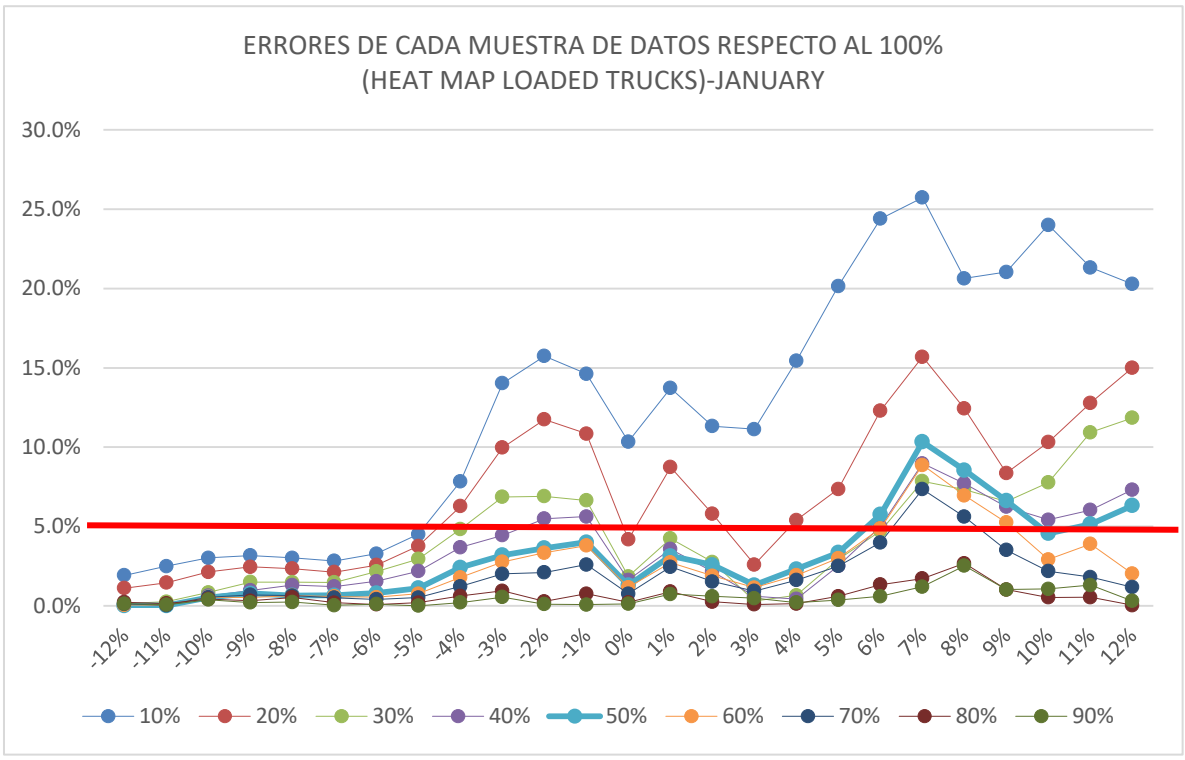
Fuente: Elaboración propia

Figura N° 22: Gráfica de los errores de cada muestra de datos para el mes de enero para camiones vacíos (Línea horizontal de color rojo representa el límite del 5% de error)



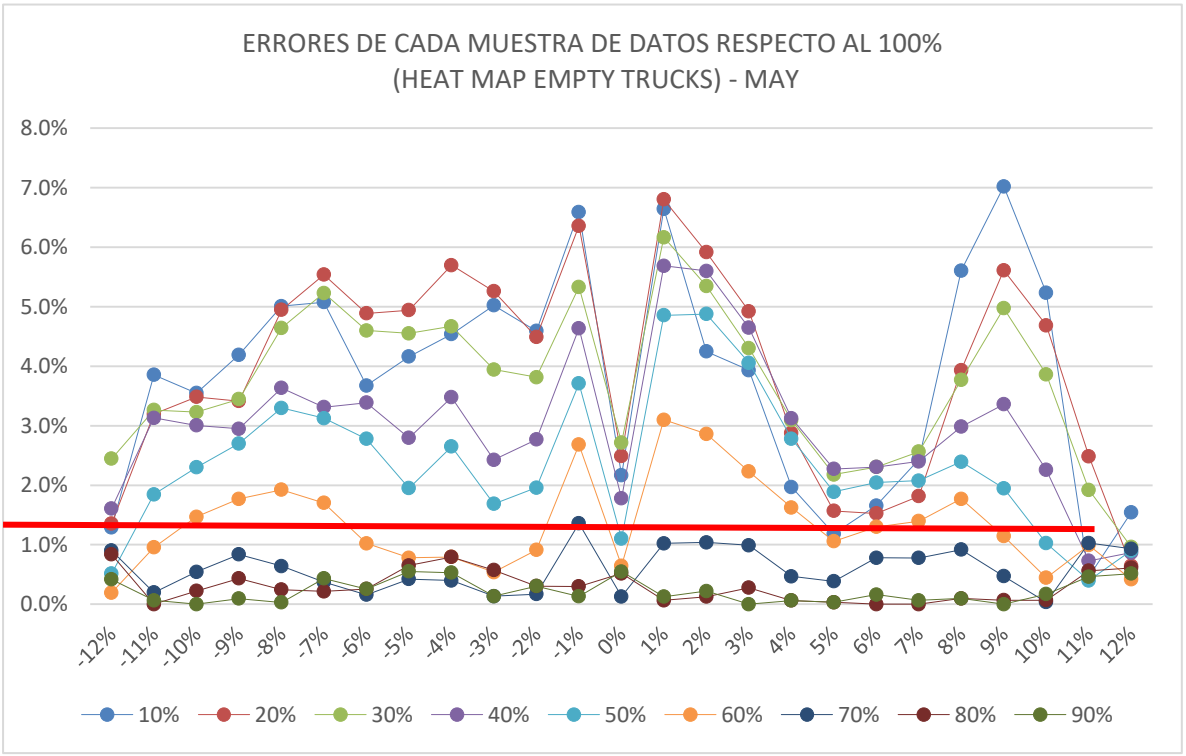
Fuente: Elaboración propia

Figura N° 23: Gráfica de errores de cada muestra de datos para el mes de enero para camiones cargados. (Línea horizontal de color rojo representa el límite del 5% de error)



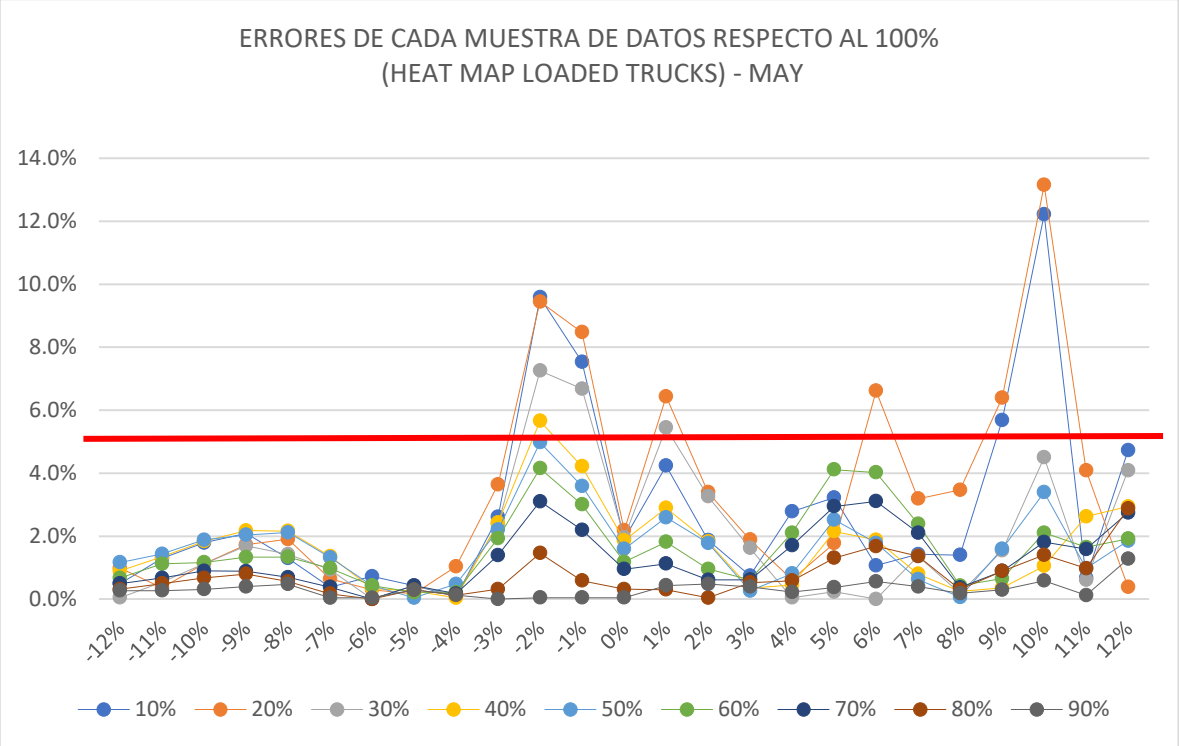
Fuente: Elaboración propia

Figura N° 24: Gráfica de errores de cada muestra de datos para el mes de mayo para camiones vacíos. (Línea horizontal de color rojo representa el límite del 5% de error)



Fuente: Elaboración propia

Figura N° 25: Gráfica de errores de cada muestra de datos para el mes de mayo para camiones cargados. (Línea horizontal de color rojo representa el límite del 5% de error)



Fuente: Elaboración propia

Figura N° 26: Mapa de calor para la flota de camiones de acarreo vacíos para el mes de enero

GRADIENT	10%	20%	30%	40%	50%	60%	70%	80%	90%
-12%	11.7%	10.2%	8.2%	6.8%	4.9%	3.8%	4.5%	3.1%	1.5%
-11%	10.8%	8.6%	7.2%	5.6%	4.3%	2.9%	3.4%	1.6%	0.6%
-10%	12.2%	7.9%	6.8%	5.6%	4.3%	3.4%	3.3%	1.3%	0.6%
-9%	13.3%	9.0%	7.5%	6.3%	5.4%	4.4%	4.1%	2.0%	0.8%
-8%	13.5%	9.4%	6.9%	5.8%	4.9%	4.3%	3.9%	2.1%	1.1%
-7%	15.3%	10.2%	6.9%	5.3%	4.4%	3.7%	3.7%	2.1%	1.1%
-6%	16.9%	12.1%	7.7%	6.3%	5.2%	3.8%	4.1%	2.2%	1.2%
-5%	17.2%	11.6%	7.7%	5.3%	4.5%	3.4%	3.7%	1.7%	0.9%
-4%	18.0%	12.0%	8.3%	5.6%	4.4%	3.3%	3.7%	1.6%	0.8%
-3%	17.2%	11.9%	7.7%	5.8%	4.5%	3.4%	3.5%	1.7%	0.9%
-2%	14.5%	10.3%	6.8%	4.9%	3.8%	2.7%	2.8%	1.5%	0.8%
-1%	11.6%	8.4%	6.0%	3.8%	2.6%	2.1%	2.2%	1.2%	0.6%
0%	10.0%	7.2%	4.5%	3.7%	2.8%	2.3%	2.2%	1.1%	0.4%
1%	9.9%	6.7%	4.1%	3.1%	2.3%	1.7%	1.8%	0.9%	0.3%
2%	8.2%	4.9%	3.3%	2.5%	1.8%	1.4%	1.4%	0.7%	0.2%
3%	8.9%	6.3%	4.8%	3.6%	2.7%	2.1%	2.1%	1.2%	0.5%
4%	9.9%	7.4%	5.6%	4.2%	3.4%	2.6%	2.5%	1.3%	0.6%
5%	9.4%	7.1%	5.5%	4.2%	3.0%	2.4%	2.1%	1.1%	0.3%
6%	8.4%	6.7%	5.4%	4.3%	3.3%	2.4%	2.0%	1.1%	0.2%
7%	7.7%	6.2%	5.1%	4.0%	3.1%	2.3%	2.3%	1.3%	0.5%
8%	8.6%	6.9%	6.0%	5.2%	3.8%	2.9%	2.9%	1.6%	0.7%
9%	10.2%	8.5%	7.1%	5.7%	4.8%	3.6%	3.6%	2.3%	0.6%
10%	11.6%	9.6%	7.2%	5.8%	4.9%	4.1%	3.7%	2.0%	0.8%
11%	10.9%	8.9%	6.4%	4.7%	3.8%	3.4%	3.0%	1.5%	0.7%
12%	7.9%	6.1%	4.1%	3.2%	3.0%	2.6%	1.8%	1.1%	0.5%

Fuente: Elaboración propia

Figura N° 27: Mapa de calor para la flota de camiones de acarreo cargados para el mes de enero

GRADIENT	10%	20%	30%	40%	50%	60%	70%	80%	90%
-12%	1.9%	1.1%	0.1%	0.1%	0.0%	0.1%	0.1%	0.2%	0.1%
-11%	2.5%	1.5%	0.3%	0.0%	0.0%	0.0%	0.0%	0.1%	0.1%
-10%	3.0%	2.1%	0.8%	0.6%	0.5%	0.5%	0.5%	0.4%	0.4%
-9%	3.2%	2.5%	1.5%	1.0%	0.8%	0.5%	0.7%	0.3%	0.2%
-8%	3.0%	2.3%	1.5%	1.3%	0.7%	0.6%	0.6%	0.5%	0.2%
-7%	2.8%	2.1%	1.5%	1.2%	0.7%	0.6%	0.5%	0.2%	0.0%
-6%	3.3%	2.6%	2.2%	1.5%	0.8%	0.5%	0.4%	0.1%	0.1%
-5%	4.5%	3.8%	2.9%	2.2%	1.1%	0.8%	0.5%	0.2%	0.0%
-4%	7.8%	6.3%	4.8%	3.7%	2.4%	1.8%	1.2%	0.6%	0.2%
-3%	14.0%	10.0%	6.9%	4.4%	3.2%	2.8%	2.0%	0.9%	0.6%
-2%	15.7%	11.8%	6.9%	5.5%	3.6%	3.3%	2.1%	0.3%	0.1%
-1%	14.6%	10.8%	6.6%	5.6%	4.0%	3.8%	2.6%	0.8%	0.1%
0%	10.3%	4.2%	1.8%	1.6%	1.3%	1.2%	0.8%	0.2%	0.1%
1%	13.7%	8.7%	4.2%	3.6%	3.1%	2.7%	2.5%	0.9%	0.7%
2%	11.3%	5.8%	2.8%	2.2%	2.6%	1.9%	1.5%	0.3%	0.6%
3%	11.1%	2.6%	0.3%	0.6%	1.3%	1.1%	0.9%	0.1%	0.5%
4%	15.4%	5.4%	0.7%	0.4%	2.3%	1.9%	1.6%	0.1%	0.2%
5%	20.2%	7.4%	2.9%	2.5%	3.4%	3.0%	2.5%	0.6%	0.4%
6%	24.4%	12.3%	4.7%	5.0%	5.8%	4.9%	4.0%	1.3%	0.6%
7%	25.7%	15.7%	7.9%	9.0%	10.3%	8.9%	7.3%	1.7%	1.2%
8%	20.6%	12.4%	7.3%	7.7%	8.5%	7.0%	5.6%	2.7%	2.5%
9%	21.0%	8.4%	6.6%	6.2%	6.6%	5.2%	3.5%	1.0%	1.0%
10%	24.0%	10.3%	7.8%	5.4%	4.6%	2.9%	2.2%	0.5%	1.1%
11%	21.3%	12.8%	10.9%	6.0%	5.1%	3.9%	1.8%	0.5%	1.3%
12%	20.3%	15.0%	11.9%	7.3%	6.3%	2.0%	1.2%	0.0%	0.3%

Fuente: Elaboración propia

Figura N°28: Mapa de calor para la flota de camiones de acarreo vacíos para el mes de mayo

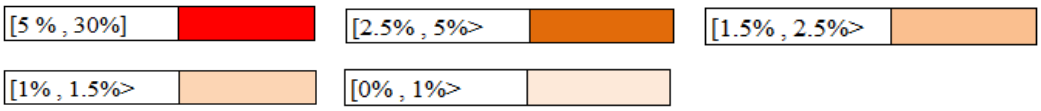
GRADIENT	10%	20%	30%	40%	50%	60%	70%	80%	90%
-12%	1.3%	1.4%	2.4%	1.6%	0.5%	0.2%	0.9%	0.8%	0.4%
-11%	3.9%	3.2%	3.3%	3.1%	1.8%	1.0%	0.2%	0.0%	0.1%
-10%	3.5%	3.5%	3.2%	3.0%	2.3%	1.5%	0.5%	0.2%	0.0%
-9%	4.2%	3.4%	3.4%	2.9%	2.7%	1.8%	0.8%	0.4%	0.1%
-8%	5.0%	4.9%	4.6%	3.6%	3.3%	1.9%	0.6%	0.2%	0.0%
-7%	5.1%	5.5%	5.2%	3.3%	3.1%	1.7%	0.4%	0.2%	0.4%
-6%	3.7%	4.9%	4.6%	3.4%	2.8%	1.0%	0.2%	0.3%	0.3%
-5%	4.2%	4.9%	4.6%	2.8%	2.0%	0.8%	0.4%	0.7%	0.6%
-4%	4.5%	5.7%	4.7%	3.5%	2.6%	0.8%	0.4%	0.8%	0.5%
-3%	5.0%	5.3%	3.9%	2.4%	1.7%	0.5%	0.1%	0.6%	0.1%
-2%	4.6%	4.5%	3.8%	2.8%	2.0%	0.9%	0.2%	0.3%	0.3%
-1%	6.6%	6.4%	5.3%	4.6%	3.7%	2.7%	1.4%	0.3%	0.1%
0%	2.2%	2.5%	2.7%	1.8%	1.1%	0.6%	0.1%	0.5%	0.5%
1%	6.6%	6.8%	6.2%	5.7%	4.9%	3.1%	1.0%	0.1%	0.1%
2%	4.2%	5.9%	5.3%	5.6%	4.9%	2.9%	1.0%	0.1%	0.2%
3%	3.9%	4.9%	4.3%	4.6%	4.1%	2.2%	1.0%	0.3%	0.0%
4%	2.0%	2.9%	3.1%	3.1%	2.8%	1.6%	0.5%	0.1%	0.1%
5%	1.2%	1.6%	2.2%	2.3%	1.9%	1.1%	0.4%	0.0%	0.0%
6%	1.7%	1.5%	2.3%	2.3%	2.0%	1.3%	0.8%	0.0%	0.2%
7%	2.4%	1.8%	2.6%	2.4%	2.1%	1.4%	0.8%	0.0%	0.1%
8%	5.6%	3.9%	3.8%	3.0%	2.4%	1.8%	0.9%	0.1%	0.1%
9%	7.0%	5.6%	5.0%	3.4%	1.9%	1.1%	0.5%	0.1%	0.0%
10%	5.2%	4.7%	3.9%	2.3%	1.0%	0.4%	0.0%	0.1%	0.2%
11%	0.5%	2.5%	1.9%	0.7%	0.4%	1.0%	1.0%	0.6%	0.5%
12%	1.5%	0.6%	1.0%	0.9%	0.9%	0.4%	0.9%	0.6%	0.5%

Fuente: Elaboración propia

Figura N°29: Mapa de calor para la flota de camiones de acarreo cargados para el mes de mayo

GRADIENT	10%	20%	30%	40%	50%	60%	70%	80%	90%
-12%	0.5%	1.0%	0.0%	0.9%	1.2%	0.7%	0.5%	0.3%	0.3%
-11%	1.3%	0.4%	0.5%	1.3%	1.4%	1.1%	0.7%	0.5%	0.3%
-10%	1.8%	1.1%	1.1%	1.8%	1.9%	1.2%	0.9%	0.7%	0.3%
-9%	2.1%	1.7%	1.7%	2.2%	2.0%	1.3%	0.9%	0.8%	0.4%
-8%	1.3%	1.9%	1.4%	2.2%	2.1%	1.3%	0.7%	0.6%	0.5%
-7%	0.4%	0.6%	0.9%	1.4%	1.3%	1.0%	0.4%	0.2%	0.0%
-6%	0.7%	0.3%	0.0%	0.3%	0.4%	0.4%	0.0%	0.0%	0.0%
-5%	0.4%	0.2%	0.4%	0.3%	0.0%	0.2%	0.4%	0.3%	0.3%
-4%	0.1%	1.0%	0.1%	0.0%	0.5%	0.2%	0.2%	0.1%	0.1%
-3%	2.6%	3.6%	2.4%	2.4%	2.2%	1.9%	1.4%	0.3%	0.0%
-2%	9.6%	9.4%	7.3%	5.7%	5.0%	4.2%	3.1%	1.5%	0.0%
-1%	7.5%	8.5%	6.7%	4.2%	3.6%	3.0%	2.2%	0.6%	0.0%
0%	1.9%	2.2%	2.0%	1.9%	1.6%	1.2%	1.0%	0.3%	0.0%
1%	4.2%	6.4%	5.4%	2.9%	2.6%	1.8%	1.1%	0.3%	0.4%
2%	1.9%	3.4%	3.3%	1.8%	1.8%	1.0%	0.6%	0.0%	0.5%
3%	0.7%	1.9%	1.6%	0.4%	0.3%	0.6%	0.6%	0.5%	0.4%
4%	2.8%	0.6%	0.0%	0.4%	0.8%	2.1%	1.7%	0.6%	0.2%
5%	3.2%	1.8%	0.2%	2.1%	2.5%	4.1%	2.9%	1.3%	0.4%
6%	1.1%	6.6%	0.0%	1.9%	1.8%	4.0%	3.1%	1.7%	0.6%
7%	1.4%	3.2%	1.4%	0.8%	0.6%	2.4%	2.1%	1.4%	0.4%
8%	1.4%	3.5%	0.2%	0.2%	0.1%	0.4%	0.4%	0.3%	0.2%
9%	5.7%	6.4%	1.5%	0.4%	1.6%	0.7%	0.9%	0.9%	0.3%
10%	12.2%	13.2%	4.5%	1.1%	3.4%	2.1%	1.8%	1.4%	0.6%
11%	0.7%	4.1%	0.6%	2.6%	1.0%	1.6%	1.6%	1.0%	0.1%
12%	4.7%	0.4%	4.1%	2.9%	1.9%	1.9%	2.7%	2.9%	1.3%

Leyenda:



Fuente: Elaboración propia

4. Discusión

Lo que este trabajo busca destacar al lector es la importancia de considerar la variabilidad que se tiene al trabajar con parte de una base de datos y la sobrestimación o subestimación que esta puede producir en cálculos posteriores que están relacionados directamente al procedimiento del cálculo de perfil de velocidades.

Dentro de las operaciones mineras se tiene como tolerancia de error en los cálculos un 5%, en otras palabras, se debe tener un grado de confiabilidad mínimo de 95%. En base al análisis de sensibilidad podemos notar a partir de qué porcentaje de muestra se puede considerar adecuado el cálculo del perfil de velocidades. Los porcentajes de la data con la cual se obtiene un grado de confiabilidad del 95% son, en el caso de enero (temporada húmeda), a partir del 60% de la data para los camiones vacíos y a partir del 80% de la data para los camiones cargados. Asimismo, en el caso de mayo (temporada seca), un grado de confiabilidad del 95%, se da a partir del 60% de la data para ambos casos de camiones vacíos y cargados.

Determinar los casos para los cuales se tiene la confiabilidad deseada en base a un análisis de sensibilidad es vital para estimar las velocidades requeridas, por ejemplo, en la elaboración del Plan de corto plazo, ya que sobreestimar o subestimar las velocidades podría conllevar a errores en la estimación de tiempos de acarreo y por ende del cálculo de la flota óptima y los costos involucrados.

Se puede concluir, además que en temporada de lluvias se tiene que prestar mayor atención al considerar con qué grado de confiabilidad se realizarán las estimaciones posteriores, para ambos casos los autores recomiendan tomar como mínimo un 80% de la data para trabajar los cálculos.

Los resultados mostrados toman en cuenta la mediana del perfil de velocidades para cada gradiente, queda pendiente aún analizar la moda y la media, pues se recomienda analizar la distribución de estas por cada gradiente y realizar una prueba de bondad de ajuste para sacar conclusiones a partir de las estadísticas y demostrar cuál de las tres medidas de tendencia central sería más representativa.

Un punto importante para poder realizar el análisis de la data con una gran cantidad de datos, es la arquitectura computacional con la que se cuenta, se recomienda un ordenador con una memoria RAM de 12 Gb de capacidad y además trabajar con el álgebra matricial para acelerar los cálculos y ahorrar costo computacional.

5. Agradecimiento

Al ingeniero Eder De La Cruz, por fomentar la investigación y el desarrollo de la programación en los jóvenes universitarios. Además de los consejos que ha brindado a este equipo para nuestro desarrollo personal y profesional.

6. Literatura Citada

De La Cruz, Eder; Aróstegui, Nelson; Huarcaya, Julio (2020). Perfil de velocidades de camiones cargados y vacíos en minería superficial.

Arnaldo Pérez Castaño (2016). Python fácil (1era. edición). Editorial MARCOMBO.

Marín Suárez, Alfredo Ph.D. (2013). Manual del Curso Taller GPSS: APLICACIÓN DE LA SIMULACIÓN DE TRANSPORTE CON GPSS AL PLANEAMIENTO DE MINADO.

7. Anexos

Código Python usado en el presente trabajo

```
from google.colab import drive
drive.mount('/content/drive')
import pandas as pd
xcoor=pd.read_excel('/content/drive/MyDrive/ColabNotebooks/GERMINA/05_Mayo/2019-05.xlsx',sheet_name='xcoor')
ycoor=pd.read_excel('/content/drive/MyDrive/ColabNotebooks/GERMINA/05_Mayo/2019-05.xlsx',sheet_name='ycoor')
tonelaje=pd.read_excel('/content/drive/MyDrive/ColabNotebooks/GERMINA/05_Mayo/2019-05.xlsx',sheet_name='tonelaje')
incl_pitch=pd.read_excel('/content/drive/MyDrive/ColabNotebooks/GERMINA/05_Mayo/2019-05.xlsx',sheet_name='incl_pitch')
for i in range(0,xcoor.shape[0]):
    elemento=xcoor.values[i][0]
    elemento=elemento.replace('{','')
    elemento=elemento.replace('}','')
```

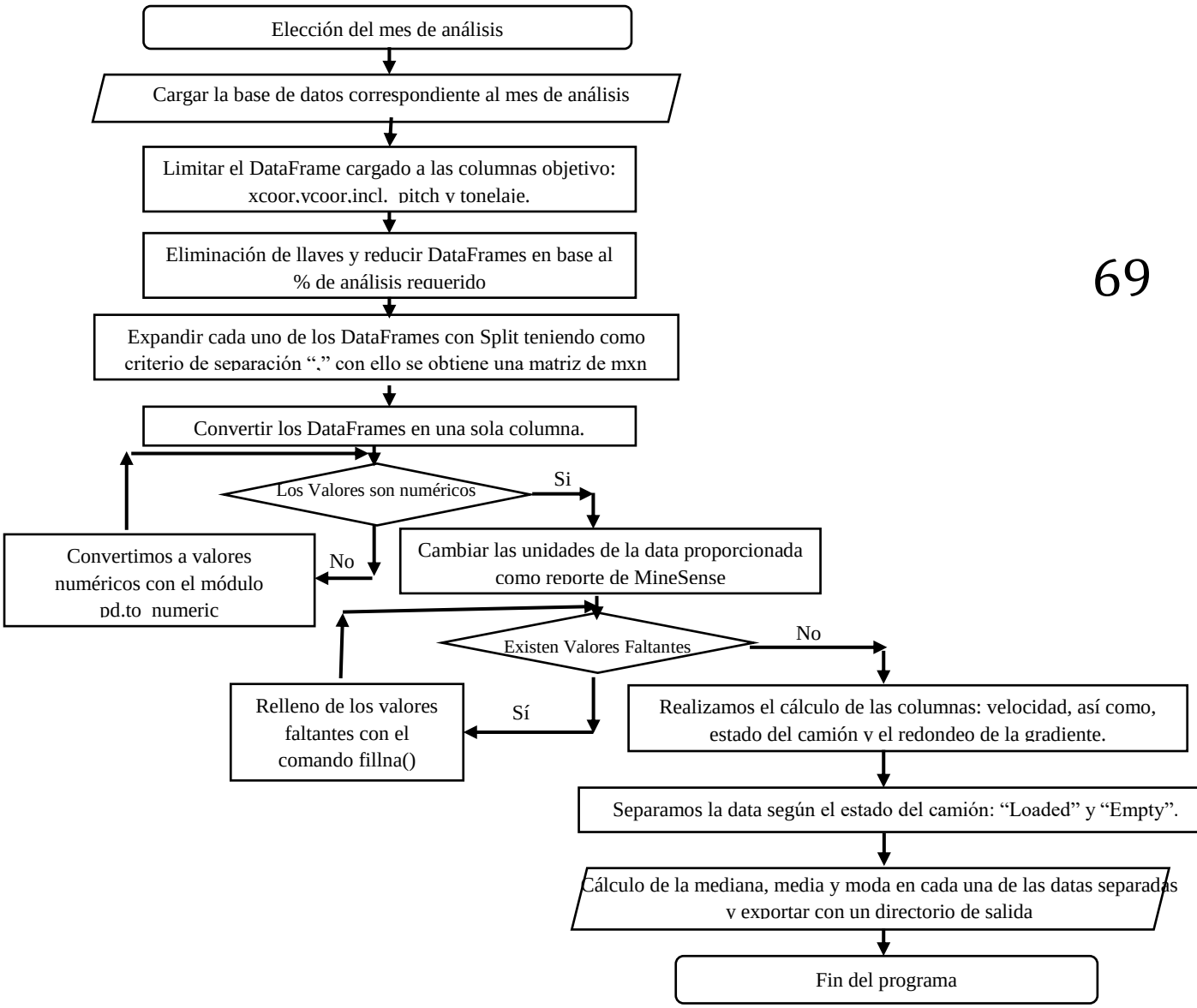
```
    #elemento=elemento.replace("","")
    xcoor.values[i][0]=elemento
for i in range(0,ycoor.shape[0]):
    elemento=ycoor.values[i][0]
    elemento=elemento.replace('{',")
    elemento=elemento.replace('}',")
    #elemento=elemento.replace("","")
    ycoor.values[i][0]=elemento
for i in range(0,tonelaje.shape[0]):
    elemento=tonelaje.values[i][0]
    elemento=elemento.replace('{',")
    elemento=elemento.replace('}',")
    #elemento=elemento.replace("","")
    tonelaje.values[i][0]=elemento
for i in range(0,incl_pitch.shape[0]):
    elemento=incl_pitch.values[i][0]
    elemento=elemento.replace('{',")
    elemento=elemento.replace('}',")
    #elemento=elemento.replace("","")
    incl_pitch.values[i][0]=elemento
elemento=0
import math
porc=1
total=xcoor.shape[0]*porc
total=math.ceil(total)
print(total)
xcoor=xcoor.loc[0:total,:]
ycoor=ycoor.loc[0:total,:]
tonelaje=tonelaje.loc[0:total,:]
incl_pitch=incl_pitch.loc[0:total,:]
xcoor=xcoor['xcoor'].str.split(",",expand=True)
ycoor=ycoor['ycoor'].str.split(",",expand=True)
tonelaje=tonelaje['tonelaje'].str.split(",",expand=True)
incl_pitch=incl_pitch['incl_pitch'].str.split(",",expand=True)
#convertimos a dataframe los xcoor en una sola columna:
xcoor=pd.DataFrame(xcoor.values.reshape(xcoor.values.shape[0]*xcoor.values.shape[1],1),col
umns=['xcoor'])
ycoor=pd.DataFrame(ycoor.values.reshape(ycoor.values.shape[0]*ycoor.values.shape[1],1),col
umns=['ycoor'])
tonelaje=pd.DataFrame(tonelaje.values.reshape(tonelaje.values.shape[0]*tonelaje.values.shape[
1],1),columns=['tonelaje'])
incl_pitch=pd.DataFrame(incl_pitch.values.reshape(incl_pitch.values.shape[0]*incl_pitch.value
s.shape[1],1),columns=['incl_pitch'])
#convertimos a valores float todos los valores
xcoor=pd.to_numeric(xcoor['xcoor'],errors='coerce')
ycoor=pd.to_numeric(ycoor['ycoor'],errors='coerce')
#tonelaje=pd.to_numeric(tonelaje['tonelaje'],errors='coerce')
incl_pitch=pd.to_numeric(incl_pitch['incl_pitch'],errors='coerce')
xcoor=pd.DataFrame(xcoor,columns=['xcoor'])
ycoor=pd.DataFrame(ycoor,columns=['ycoor'])
#tonelaje=pd.DataFrame(tonelaje,columns=['tonelaje'])
incl_pitch=pd.DataFrame(incl_pitch,columns=['incl_pitch'])
xcoor['xcoor']=xcoor['xcoor']/100
```

```
ycoor['ycoor']=ycoor['ycoor']/100
#tonelaje['tonelaje']=tonelaje['tonelaje']/10
incl_pitch['incl_pitch']=incl_pitch['incl_pitch']/100
tonelaje[(tonelaje['tonelaje'].isnull())]
ycoor[(ycoor['ycoor'].isnull())]
xcoor=xcoor.fillna(-80)
ycoor=ycoor.fillna(-80)
incl_pitch=incl_pitch.fillna(-80)
import math
valores=incl_pitch['incl_pitch'].values*(math.pi/180)
import numpy as np
valores=np.tan(valores)*100
data2=pd.DataFrame(valores,columns=['incl_pitch_%'])
#convertimos los dataframes separados a matrices
xcoor_matriz=xcoor['xcoor'].to_numpy()
ycoor_matriz=ycoor['ycoor'].to_numpy()
tonelaje_matriz=tonelaje['tonelaje'].to_numpy()
incl_pitch_matriz=incl_pitch['incl_pitch'].to_numpy()
incl_pitch_por_matriz=data2['incl_pitch_%'].to_numpy()
#remodelamos las matrices
xcoor_matriz=xcoor_matriz.reshape(xcoor.shape[0]//1800,1800)
ycoor_matriz=ycoor_matriz.reshape(ycoor.shape[0]//1800,1800)
tonelaje_matriz=tonelaje_matriz.reshape(tonelaje_matriz.shape[0]//1800,1800)
incl_pitch_matriz=incl_pitch_matriz.reshape(incl_pitch.shape[0]//1800,1800)
incl_pitch_por_matriz=incl_pitch_por_matriz.reshape(incl_pitch.shape[0]//1800,1800)
long=(xcoor_matriz.shape[1]-1)*xcoor_matriz.shape[0]
velocidades=np.zeros(long)
velocidades=velocidades.reshape(1799,long//1799)
gradientes=np.zeros(long)
gradientes=gradientes.reshape(1799,long//1799)
estado=np.zeros(long)
estado=estado.reshape(1799,long//1799)
m=0
n=xcoor_matriz.shape[1]-1 #1799
k=xcoor_matriz.shape[0]   #10000
for i in range(m,n):
    d=((xcoor_matriz.T[i]-xcoor_matriz.T[i+1])**2+(ycoor_matriz.T[i]-
ycoor_matriz.T[i+1])**2)**(0.5)
    v=(d/np.cos(incl_pitch_matriz.T[i]*math.pi/180))*0.5*3.6
    velocidades[i]=v
    ##############lista_velocidades.append(v)
    #ton=np.zeros(k)
    ton_bool=tonelaje_matriz.T[i]=="0"
    #for j in range(0,k):
        #if ton_bool[j]==True:
            #ton[j]=0
        #else:
            #ton[j]=1
    estado[i]=ton_bool
    ############################
    incl=np.zeros(k)
    x=incl_pitch_por_matriz.T[i]
    incl_bool=np.ceil(x)-x>=x-np.floor(x)
```

```
    for j in range(0,k):
        if incl_bool[j]==True:
            incl[j]=np.floor(x[j])
        else:
            incl[j]=np.ceil(x[j])
    gradientes[i]=incl.
#remodelamos las matrices
velocidades=velocidades.T.reshape(long,1)
gradientes=gradientes.T.reshape(long,1)
estado=estado.T.reshape(long,1)
df_v=pd.DataFrame(velocidades,columns=['velocidad'])
df_g=pd.DataFrame(gradientes,columns=['gradiente'])
df_e=pd.DataFrame(estado,columns=['estado'])
data=pd.concat([df_v,df_g,df_e],axis=1)
data_vacio=data.loc[(data['estado']==True)]
data_cargado=data.loc[(data['estado']==False)]
lista_gradientes=[]
for i in range(-12,13):
    lista_gradientes.append(i)
v_media=[]
v_moda=[]
v_mediana=[]
for i in lista_gradientes:
    df_g=data_cargado.loc[(data_cargado['gradiente'])==i]
    df_g=df_g.loc[(df_g['velocidad'])>=10]
    df_g=df_g.loc[(df_g['velocidad'])<=65]
    v_media_g=round(df_g['velocidad'].mean(),2)
    v_mediana_g=round(df_g['velocidad'].median(),2)
    v_moda_g=round(df_g['velocidad'].mode()[0],2)
    v_media.append(v_media_g)
    v_mediana.append(v_mediana_g)
    v_moda.append(v_moda_g)

dicc = {
    "gradientes": lista_gradientes,
    "v_media": v_media,
    "v_moda":v_moda,
    "v_mediana":v_mediana
}

df = pd.DataFrame(dicc)
print(df)
df.to_excel('mayo_100.xlsx')
v_media=[]
v_moda=[]
v_mediana=[]
for i in lista_gradientes:
    df_g=data_vacio.loc[(data_vacio['gradiente'])==i]
    df_g=df_g.loc[(df_g['velocidad'])>=10]
    df_g=df_g.loc[(df_g['velocidad'])<=65]
    v_media_g=round(df_g['velocidad'].mean(),2)
    v_mediana_g=round(df_g['velocidad'].median(),2)
    v_moda_g=round(df_g['velocidad'].mode()[0],2)
```

```
    v_media.append(v_media_g)
    v_mediana.append(v_mediana_g)
    v_moda.append(v_moda_g)

dicc = {
    "gradientes": lista_gradientes,
    "v_media": v_media,
    "v_mediana":v_mediana,
    "v_moda":v_moda
    }

df = pd.DataFrame(dicc)
print(df)
df.to_excel('mayo100_v.xlsx')
```

Diagrama de flujo



REVISTA DE INVESTIGACIÓN MULTIDISCIPLINARIA

CTSCAFE Para ciudadanos…

http://www.ctscafe.pe

Volumen VI- N° 17 Julio 2022

Contáctenos en nuestro correo electrónico
revistactscafe@ctscafe.pe

Página Web:
http://ctscafe.pe

Blog:
https://ctscafeparaciudadanos.blogspot.com/

Facebook
https://www.facebook.com/Revista-CTSCafe-1822923591364746/

latindex