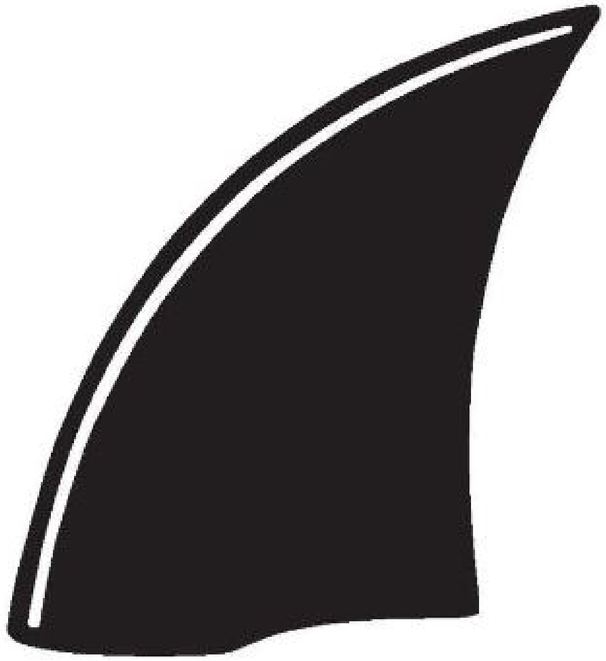
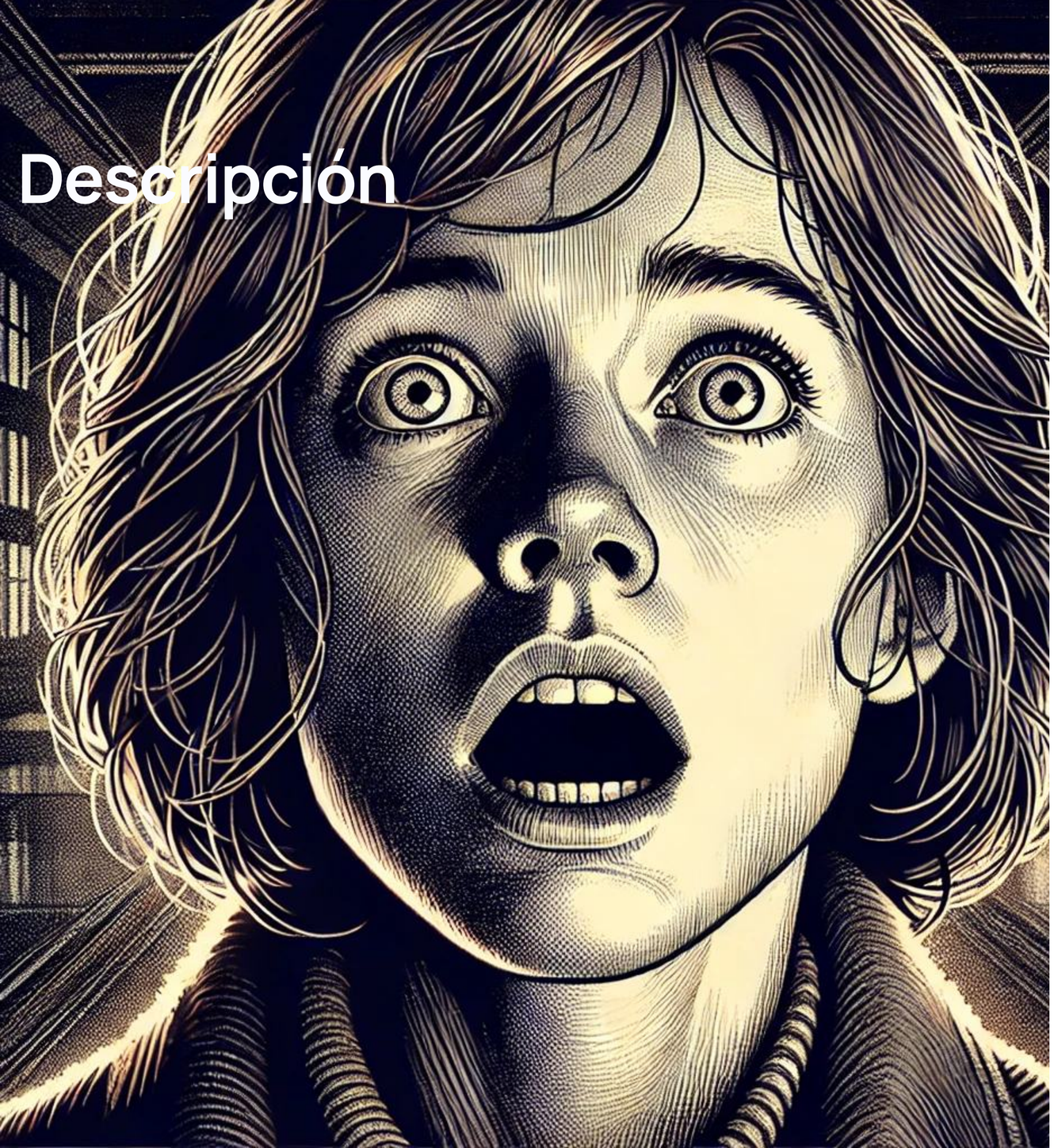
SHARK ATTACK SURVIVOR



Fernando Sanz-Extremera I Laura Nieto I Elisa Correa

Descripción

Somos una ONG dedicada a prevención de ataques de tiburones. Analizamos las **zonas** con **mayor incidencia** y las causas principales de los ataques, con el fin de desarrollar **campañas preventivas efectivas**.



Descripción

Original Data Set

-Incidentes provocados por ataques de tiburones

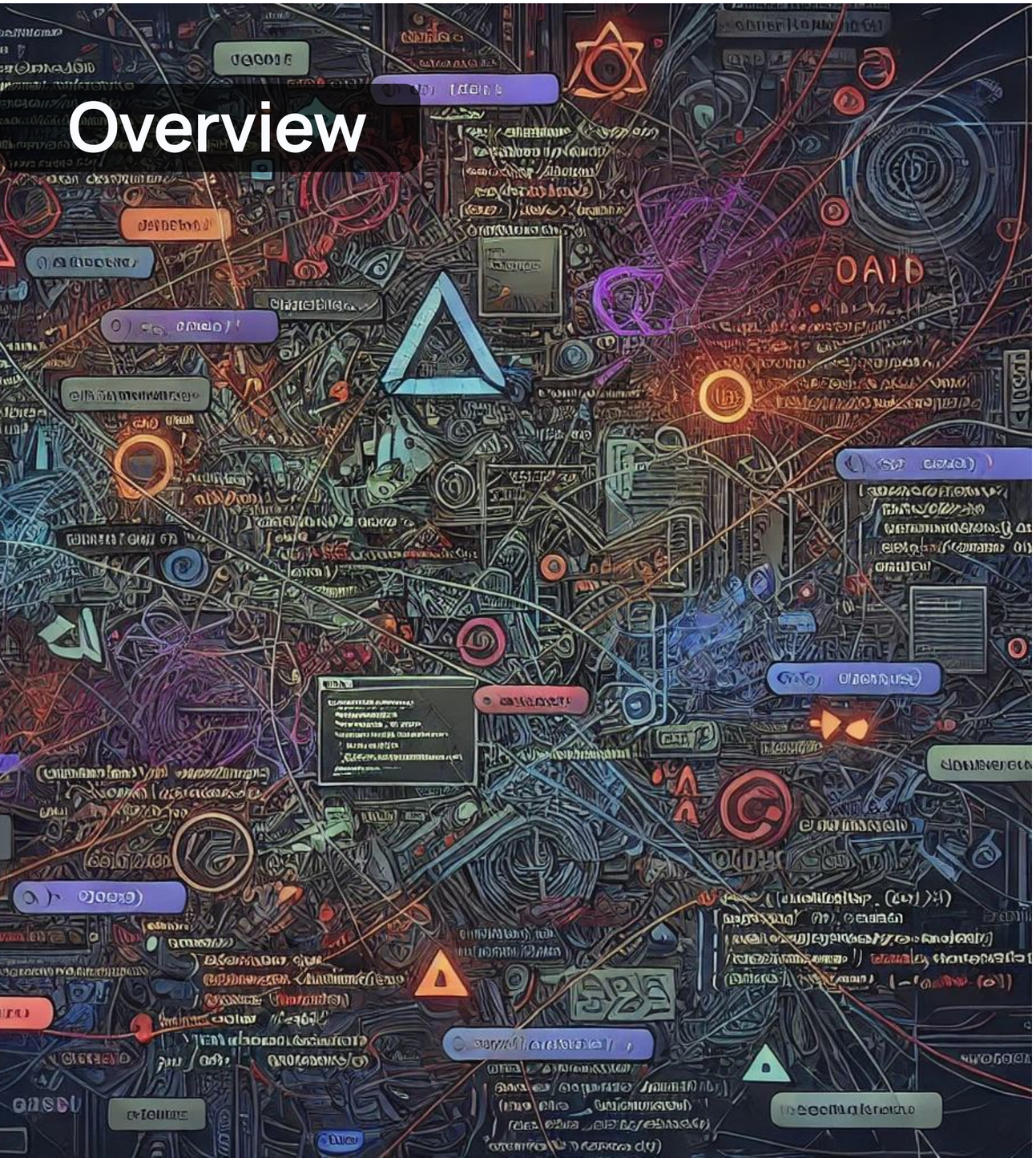
-1853-2024

-Tipo de especies

-Actividad

-Nombres, sexo y edad

-Fuentes y archivos



Proceso de limpieza

- Columnas
- Filas duplicadas
- Quitar NaNs
- Minúscula y espacios.
- Rellenar NaNs
- Agrupar

Retos y soluciones

- Limpiar
- Agrupar
- Filtrar
- Reemplazar

Limpiamos | NaNs , espacios y unificamos minúsculas

[19]:
```
# Quitamos las filas que tienen más de 3 NaNs
clean_df = clean_df.dropna(thresh = 4)
clean_df
```

[25]:
```
# Rellenamos los nans de type con Unknown
clean_df.type = clean_df.type.fillna("Unknown")
```

[48]:
```
# Rellenamos los 160 nans de activity con Unknown
clean_df.activity = clean_df.activity.fillna("Unknown")
```

[50]:
```
#Pasamos a minusculas y quitamos espacios al principio y final
clean_df.activity = clean_df.activity.apply(lambda x: x.lower().strip())
clean_df.activity.value_counts()
```

Limpiar y agrupar | Texto --› Regex

[393]:
```
test_df.activity = test_df.activity.str.replace(".*surf.*", "surfing", regex = True)
test_df.activity = test_df.activity.str.replace(".*swim+ing.*", "swimming", regex = True)
test_df.activity = test_df.activity.str.replace(".*fishing.*", "fishing", regex = True)
test_df.activity = test_df.activity.str.replace(".*spearfishing.*", "spearfishing", regex = True)
test_df.activity = test_df.activity.str.replace(".*div.*", "diving", regex = True)
test_df.activity = test_df.activity.str.replace(".*body.boarding.*", "bodyboarding", regex = True)
test_df.activity = test_df.activity.str.replace(".*bo.gie boarding.*", "boogie boarding", regex = True)
test_df.activity = test_df.activity.str.replace(".*snorkel.*", "snorkeling", regex = True)
test_df.activity = test_df.activity.str.replace(".*film.*", "filming", regex = True)
test_df.activity = test_df.activity.str.replace(".*photo.*", "photographing", regex = True)
test_df.activity = test_df.activity.str.replace(".*shark.*", "shark related activities", regex = True)
test_df.activity = test_df.activity.str.replace(".*play.*", "playing", regex = True)
test_df.activity = test_df.activity.str.replace(".*float.*", "floating", regex = True)
test_df.activity = test_df.activity.str.replace(".*paddl.*", "paddleboarding", regex = True)
test_df.activity = test_df.activity.str.replace(".*fell.*", "swimming", regex = True)
test_df.activity = test_df.activity.str.replace(".*jump.*", "swimming", regex = True)
test_df.activity = test_df.activity.str.replace(".*stand.*", "swimming", regex = True)
test_df.activity.value_counts()
```

Limpiar y agrupar | por Categorías

[363]:
```
test2_df.country = test2_df.country.replace("columbia", "colombia")
test2_df.country = test_df.country.replace(["french polynesia", "new caledonia", "reunion", "reunion island", "st martin"], "french over
test2_df.country = test_df.country.replace(["british overseas territory", "british virgin islands", "cayman islands", "diego garcia", "e
test2_df.country = test2_df.country.replace(["puerto rico", "guam"], "usa")
test2_df.country = test2_df.country.replace(["aruba", "st. maartin"], "netherlands")
test2_df.country = test2_df.country.replace("azores", "portugal")
test2_df.country = test2_df.country.replace("hong kong", "china")
test2_df.country = test2_df.country.replace("coral sea", "australia")
test2_df.country = test2_df.country.replace("okinawa", "japan")
test2_df.country = test2_df.country.replace(["palestinian territories", "egypt / israel"], "israel")
test2_df.country = test2_df.country.replace("united arab emirates (uae)", "united arab emirates")
test2_df.country = test2_df.country.replace("antigua", "Antigua and Barbuda")
test2_df.country = test2_df.country.replace("maldive islands", "maldives")
test2_df.country = test2_df.country.replace("tobago", "Trinidad and Tobago")
test2_df.country = test2_df.country.replace(["nevis", "st kitts / nevis"], "Saint Kitts and Nevis")
test2_df.country = test2_df.country.replace(["atlantic ocean", "caribbean sea", "gulf of aden", "northern arabian sea"], "middle of the
test2_df.country = test2_df.country.replace("nan", "unknown")
```

[399]:
```
")
 "removing fish from a trap", "scalloping", "shrimping", "picking opihi"], "other fishing activities")
", "lying prone in 2' of water", "crawling", "squatting in the water", "watching seals", "watching the sardine run", "sightseeing"], "rec
oting to catch a crocodile", "attempting to fix motor", "attempting to illegally enter the USA", "attempting to retrieve a dinghy"], "oth
```

Creamos un buscador | para reemplazar palabras

[672]:
```
#Buscador en injury:
list(test3_df[test3_df.injury.str.contains("abras")].injury.sort_values().unique())
```

[672]:
```
["abrasion to arm from shark's rough skin",
 'abrasion to leg when he kicked the shark provoked incident',
 'abrasion to right forearm from pectoral fin of a shark that leapt into his boat',
 'abrasions',
 'abrasions and cuts to sole of foot',
 'abrasions to elbow; collided with shark',
 'abrasions to left hand',
 'bruises and abrasions to face, chin, chest, both shins & feet and cut to right hand when her surfboard was struck with force',
 'bruises, abrasions and some spinal and nerve damage when collided with marine animal, possibly a shark or dolphin.',
 'minor abrasions to legs when she was lifted on the back of a large marine animal']
```

[674]:
```
test3_df.injury = test3_df.injury.str.replace(".*presumed fatal.*", "fatal", regex = True)
test3_df.injury = test3_df.injury.str.replace(".*bodies.*", "fatal", regex = True)
test3_df.injury = test3_df.injury.str.replace(".* fatal.*", "fatal", regex = True)
test3_df.injury = test3_df.injury.str.replace(".*fatal,.*", "fatal", regex = True)
test3_df.injury = test3_df.injury.str.replace(".*fatal .*", "fatal", regex = True)
test3_df.injury = test3_df.injury.str.replace(".*no injur.*", "no-injury", regex = True)
test3_df.injury = test3_df.injury.str.replace(".* injur.*", "injured", regex = True)
test3_df.injury = test3_df.injury.str.replace("^injur.*", "injured", regex = True)
test3_df.injury = test3_df.injury.str.replace(".*bit.*", "injured", regex = True)
test3_df.injury = test3_df.injury.str.replace(".*lace.*", "injured", regex = True)
test3_df.injury = test3_df.injury.str.replace(".*wound.*", "injured", regex = True)
test3_df.injury = test3_df.injury.str.replace(".*sever.*", "injured", regex = True)
test3_df.injury = test3_df.injury.str.replace(".*abras.*", "injured", regex = True)
test3_df.injury = test3_df.injury.str.replace(".*death.*", "fatal", regex = True)
```

Filtramos | por relevancia de información

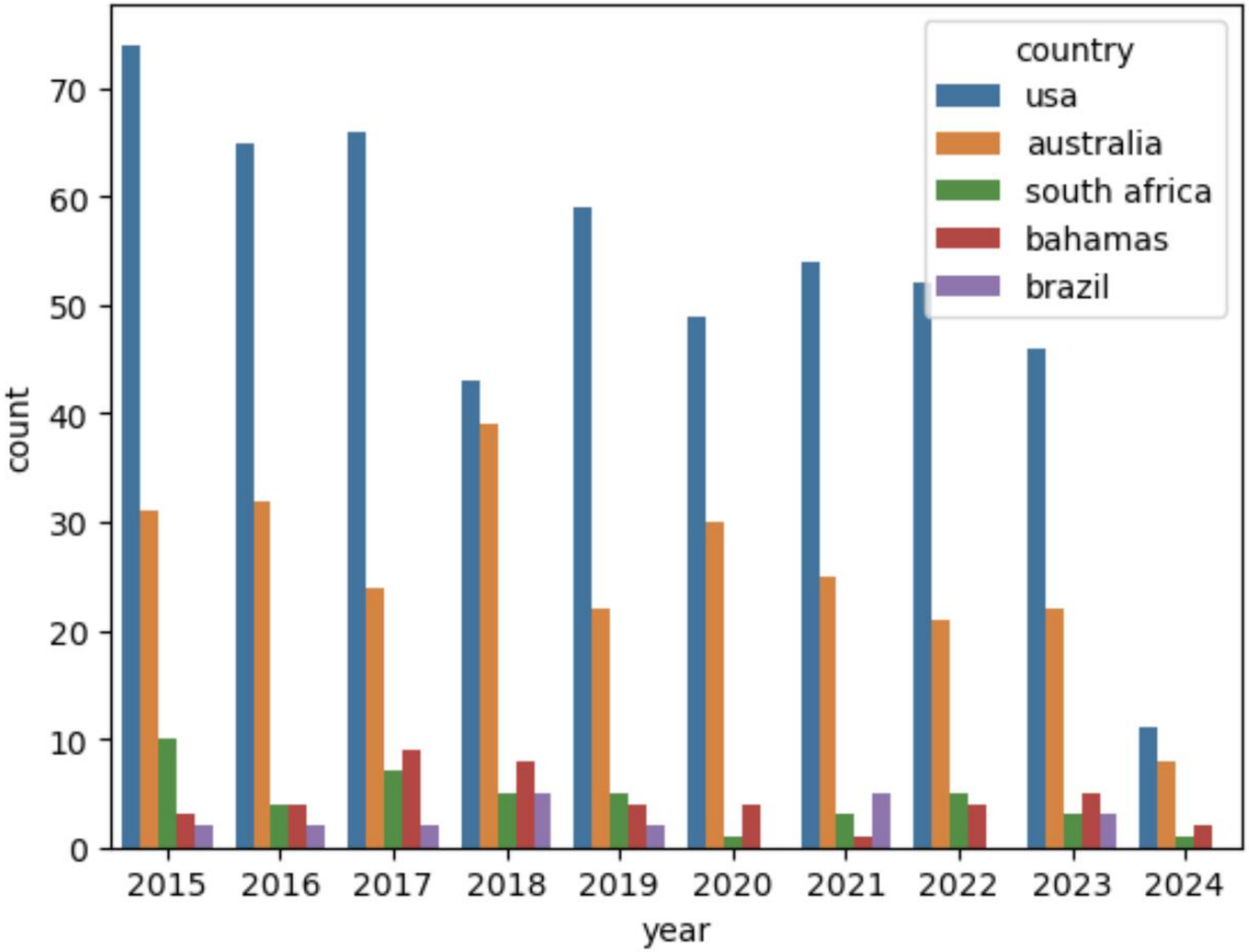
```
[43]: #Filtramos para quedarnos solo con los ataques posteriores a 1990
      clean_df = clean_df[clean_df.year > 1990]
```

Reemplazamos | por contexto

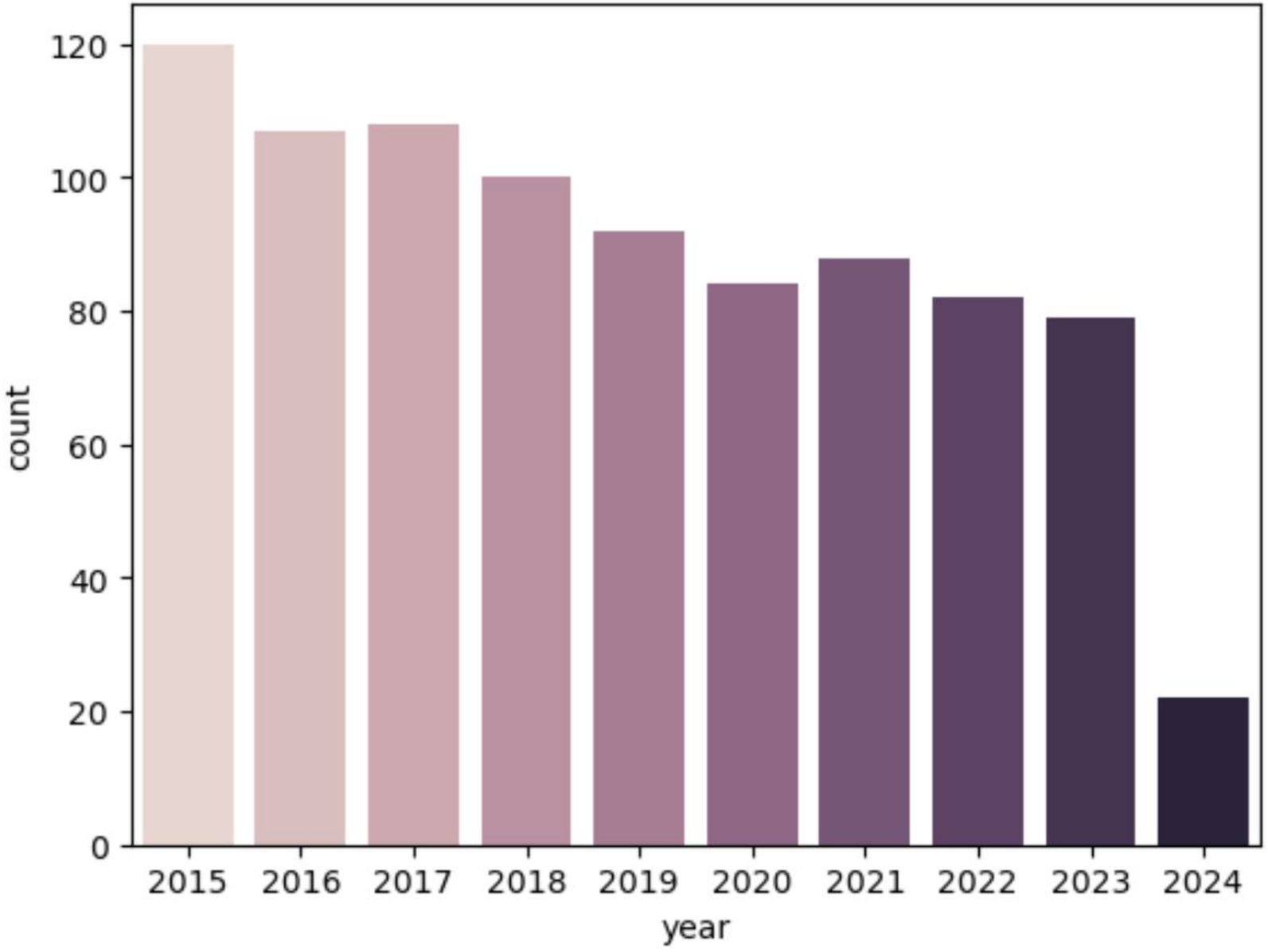
```
[363]: test2_df.country = test2_df.country.replace("columbia", "colombia")
```

Obstáculos y Aprendizajes

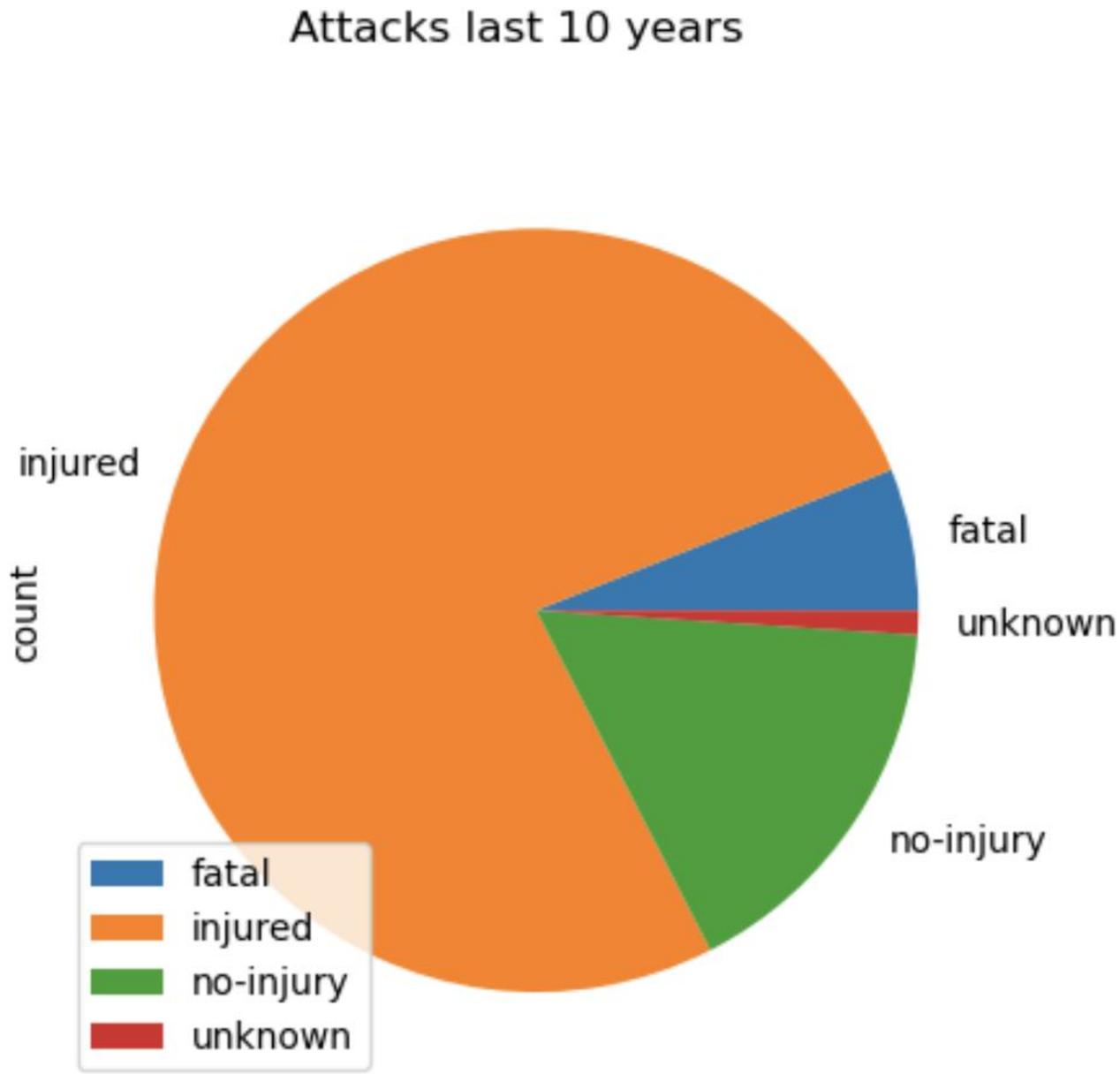
Conclusiones



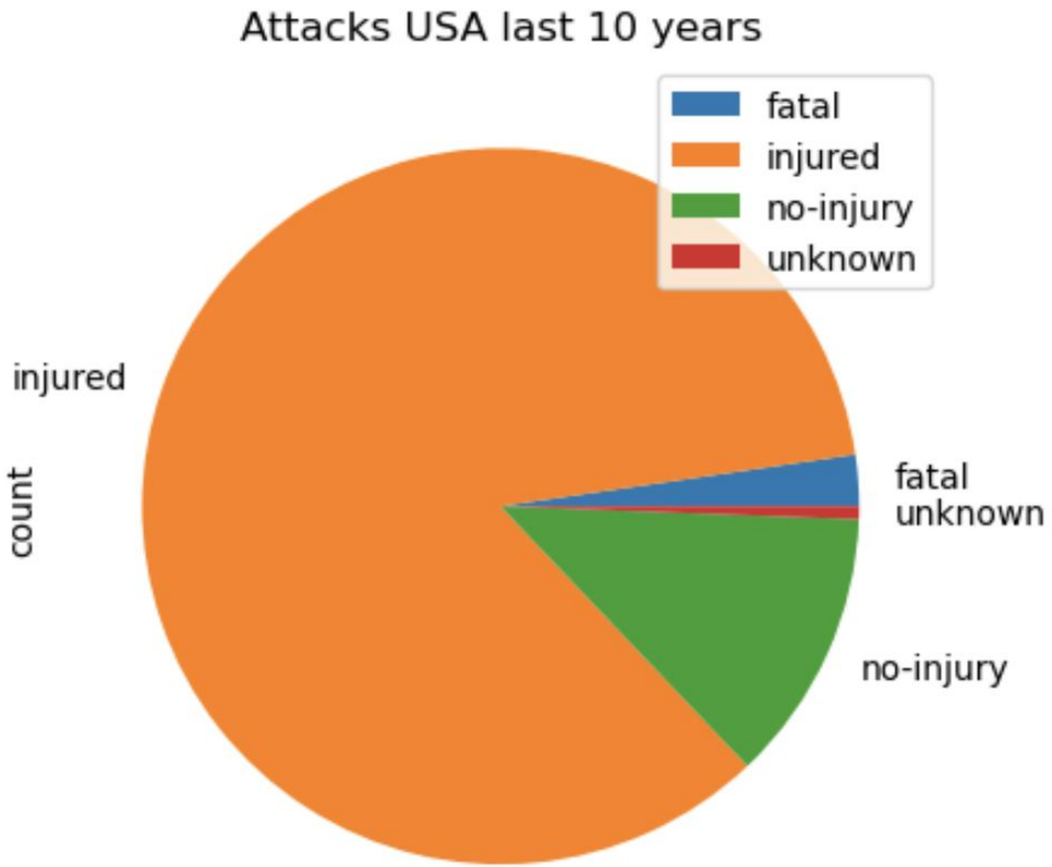
Conclusiones



Conclusiones

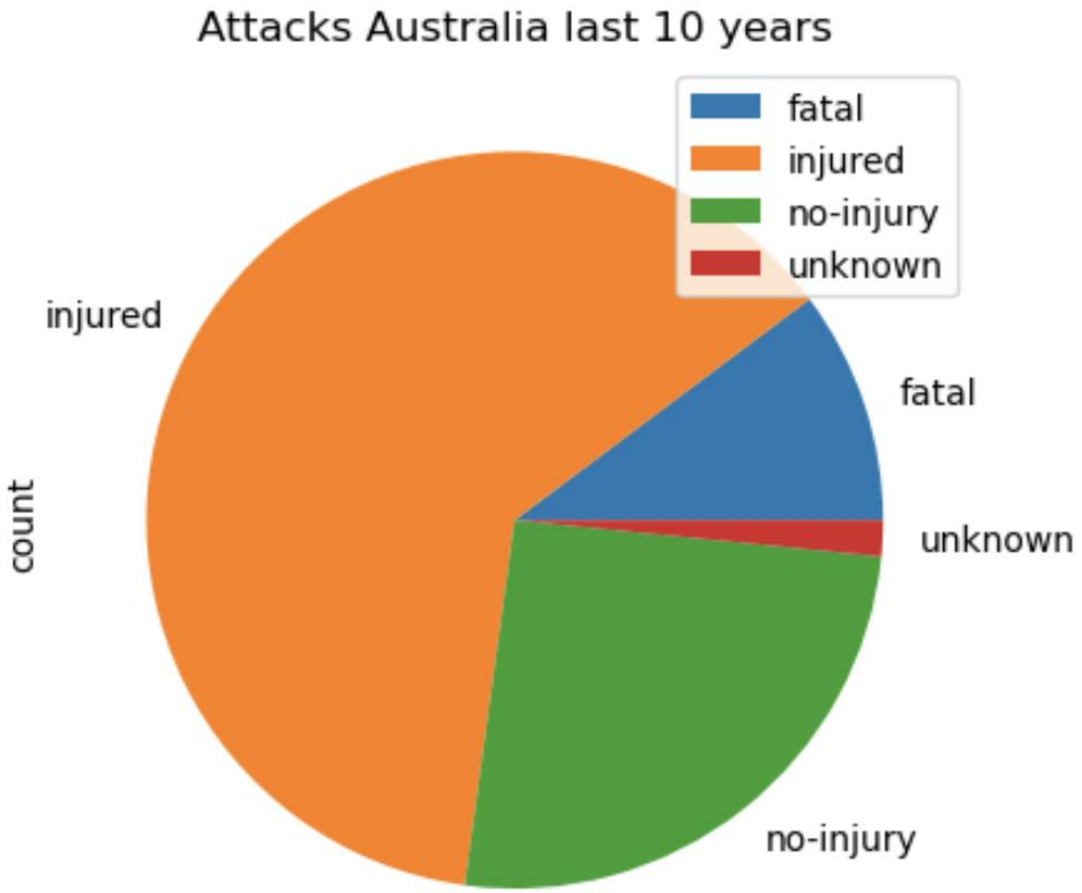


Conclusiones



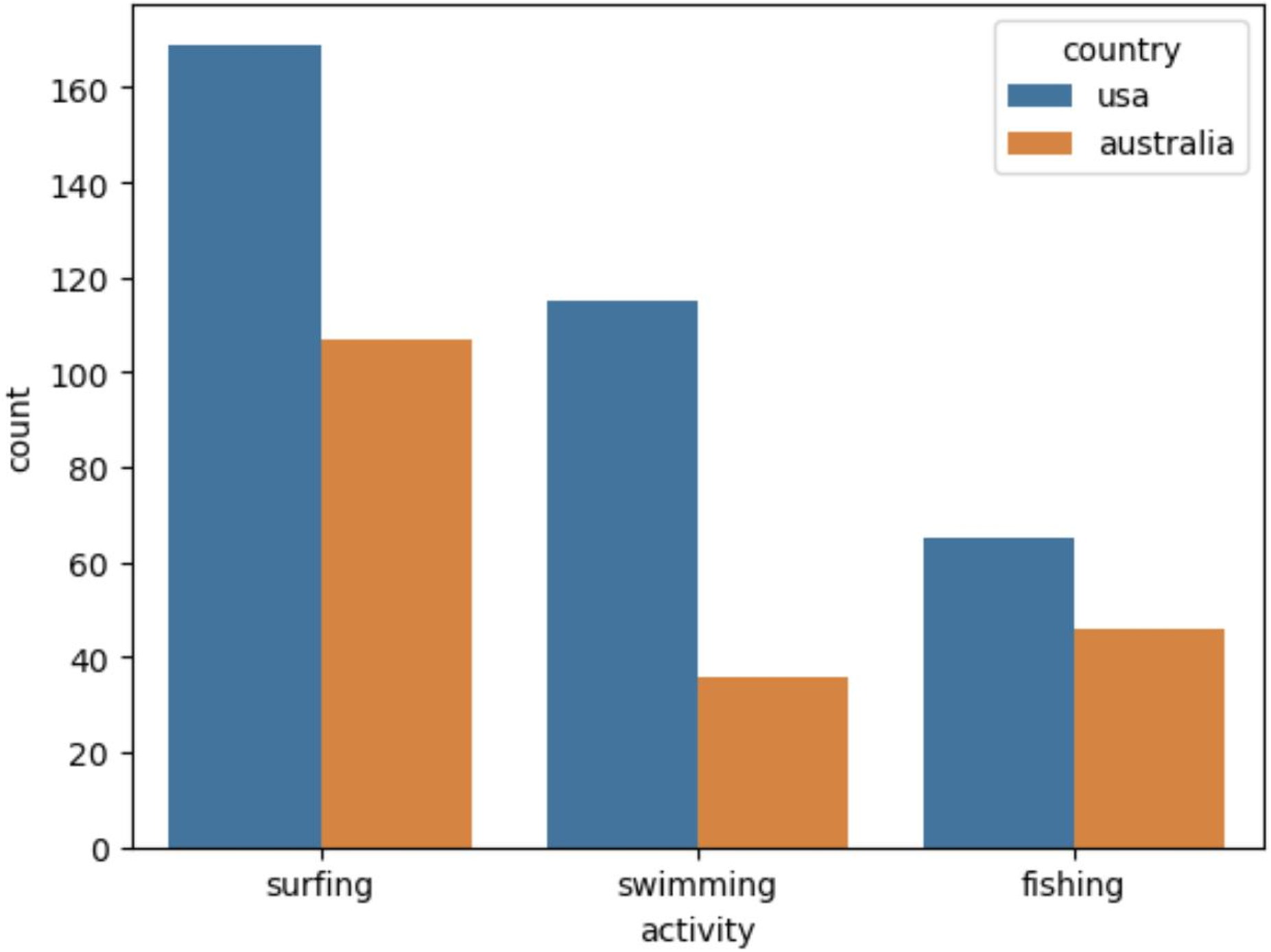
injury	count
fatal	12
injured	440
no-injury	64
unknown	3

Conclusiones



injury	count
fatal	26
injured	159
no-injury	65
unknown	4

Conclusiones



Gracias