¿Cómo decide un Data Scientist adónde viajar?

Jorge López Lázaro
Senior Data Scientist

instituto de ingeniería del conocimiento

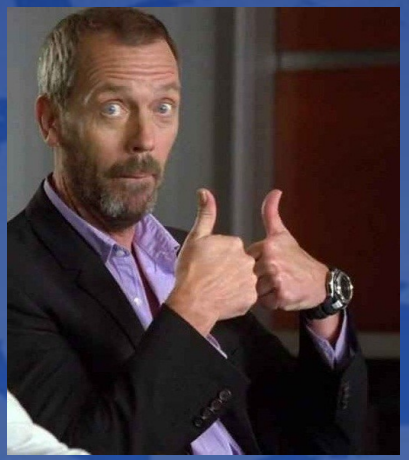
datahack

Un Data Scientist...

- Tiene la profesión más sexy del siglo XXI.
- Es alguien curioso por naturaleza → normal que le guste viajar.
- Cobra lo suficiente como para no recurrir siempre a viajes low-cost.
- Es un friki de los datos → usa Data Science para ver adónde ir...

1) Formula el problema

- Problema genérico: adónde ir.
- Problema acotado: adónde ir en avión, y volando sin escalas (a ser posible).
- Herramientas actuales para eso: Google Flights , Skyscanner...
- Ventajas:
  - Rápidas, fáciles de usar.
  - Sabes destinos y precios.
  - Desde un punto de partida fijo, guay...

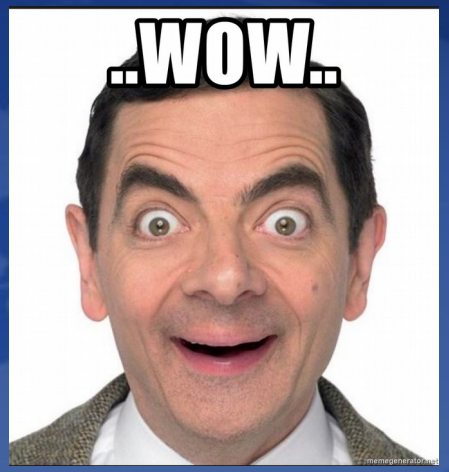


2) No se conforma con lo que hay

- Limitaciones:
  - ¿Y si no me están saliendo todas las opciones?
  - ¿Y si quiero varios vuelos parando días entremedias?
  - ¿Y si sólo me interesan las aerolíneas que me dan puntos?
  - ¿Y si no quiero que los precios varíen cada vez que consulto?

3) Descubre fuentes de datos

- Necesitaríamos vuelos mundiales...
- En directo en Flightradar24:
  - Aeropuertos.
  - Aerolíneas.
  - Aviones.
  - Vuelos actualizados.
  - ¡Pero no se pueden descargar!
- Y otros sitios ¡de pago!
  - OAG.
  - Flight Service Bureau.



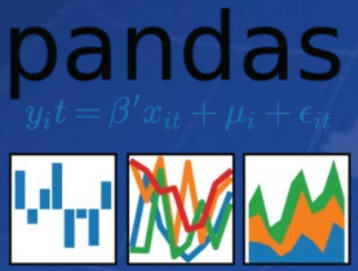
4) Pondera con cuál quedarse

- OpenFlights:
  - Aeropuertos, aerolíneas y rutas.
  - Gratis (con donación sugerida).
  - ¡Rutas desactualizadas desde 2014!

- A pesar de no ser info del todo actual:
  - La que hay es fiable.
  - Se podría cambiar después por otra.
  - Está en Kaggle y nos podemos basar en trabajo previo...

5) Elige con qué hacer el análisis

- Python con librerías open-source:
  - Leer datos con Pandas.
  - Visualizar con Holoviews+Bokeh+Leaflet.
  - Ejecutar en Jupyter notebook.



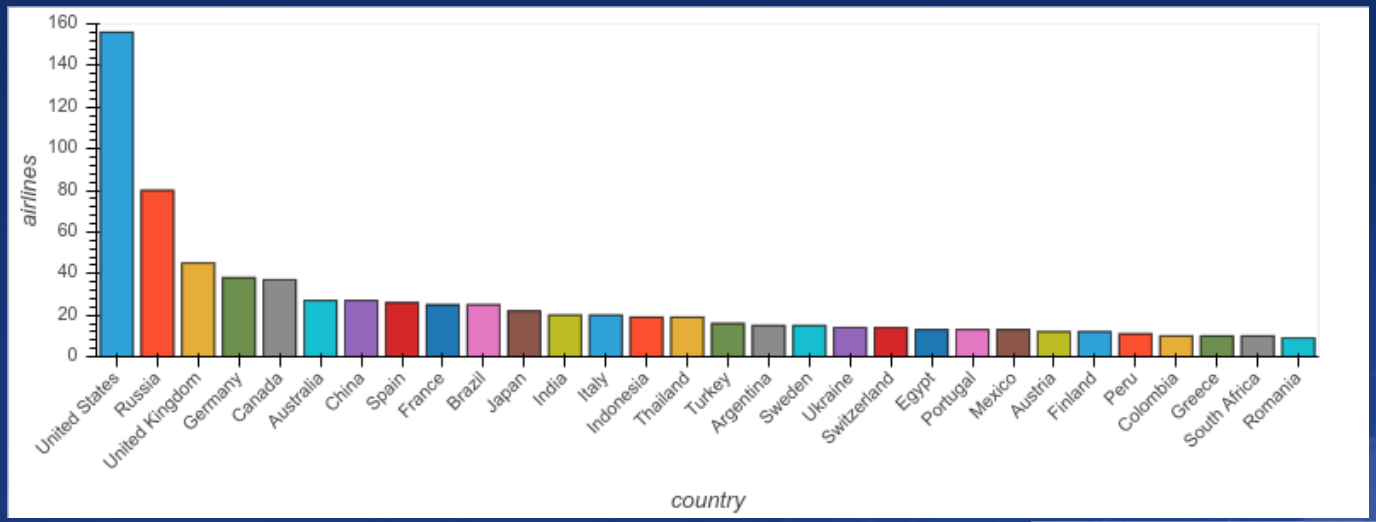
6) Procesa las aerolíneas

- 1200+ activas.
- Todas con ID, código IATA y nombre.
- De distintos países.

	Airline ID	Name	IATA	ICAO	Callsign	Country
3	3	1Time Airline	1T	RNX	NEXTIME	South Africa
10	10	40-Mile Air	Q5	MLA	MILE-AIR	United States
13	13	Ansett Australia	AN	AAA	ANSETT	Australia
14	14	Abacus International	1B	NaN	NaN	Singapore
21	21	Aigle Azur	ZI	AAF	AIGLE AZUR	France
22	22	Aloha Airlines	AQ	AAH	ALOHA	United States
24	24	American Airlines	AA	AAL	AMERICAN	United States
28	28	Asiana Airlines	OZ	AAR	ASIANA	Republic of Korea
29	29	Askari Aviation	4K	AAS	AL-AAS	Pakistan
32	32	Afriqiyah Airways	8U	AAW	AFRIQIYAH	Libya

7) Explora las aerolíneas

- ¿Países con más aerolíneas?



- ¿Aerolíneas españolas?
  - Datos algo antiguos...
  - OK para grandes.

	Airline ID	Name	IATA	ICAO	Callsign
90	90	Air Europa	UX	AEA	EUROPA
327	327	Air Nostrum	YW	ANE	AIR NOSTRUM
817	817	Air Madrid	NM	DRD	ALADA AIR
1087	1087	Air Plus Comet	A7	MPD	RED COMET
1606	1607	Calima Aviacion	XG	CLI	CALIMA
2820	2822	Iberia Airlines	IB	IBE	IBERIA
2946	2948	Islas Airways	IF	ISW	PINTADERA
3209	3211	LTE International Airways	XO	LTE	FUN JET
3829	3834	PAN Air	PV	PNR	SKYJET
4644	4652	Spanair	JK	JKK	SPANAIR
5335	5352	Vueling Airlines	VY	VLG	VUELING
5633	11795	Andalus Lineas Aereas	NaN	ANU	Andalus
5670	12962	Gadair European Airlines	GP	GDR	GADAIR
5893	17658	Volotea	NaN	VOO	Volotea

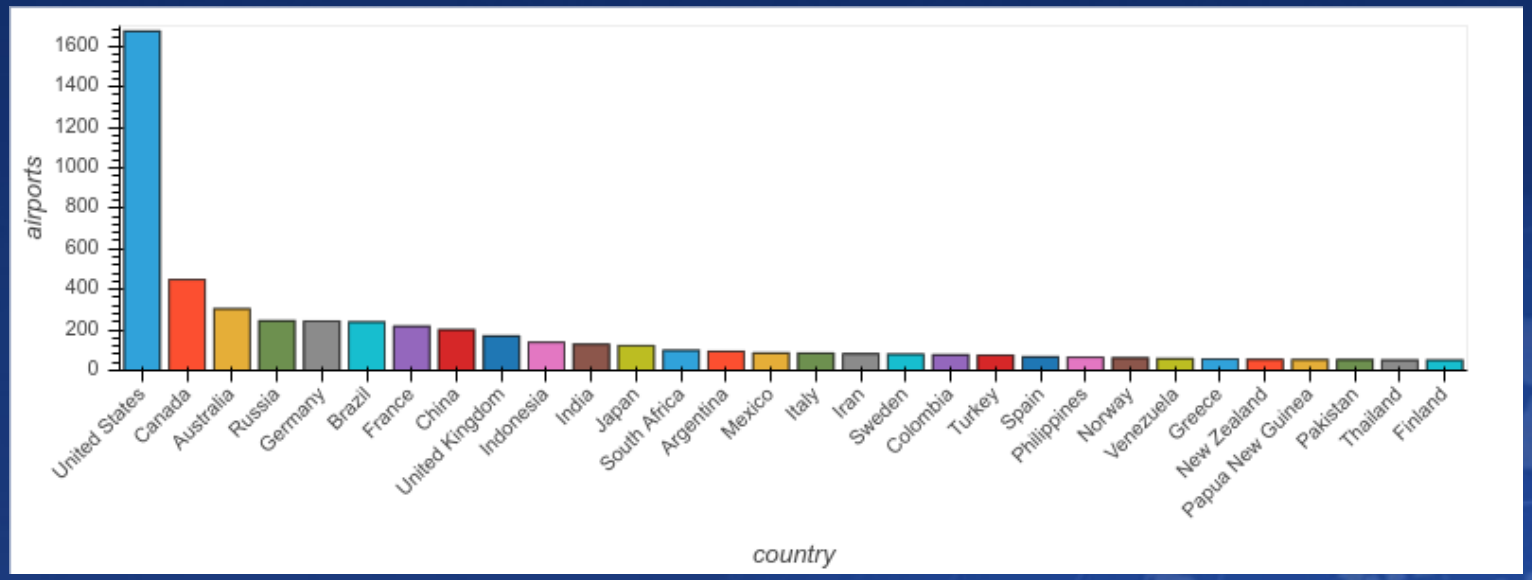
8) Procesa los aeropuertos

- 7700+ operativos.
- Todos con ID, código IATA y nombre.
- Con localización por país, ciudad, altitud e incluso coordenadas.

	Airport ID	Name	City	Country	IATA	ICAO	Latitude	Longitude	Altitude	HoursFromUTC	DST	Timezone
0	1	Goroka Airport	Goroka	Papua New Guinea	GKA	AYGA	6.081690	145.391998	5282	10.0	U	Pacific/Port_Moresby
1	2	Madang Airport	Madang	Papua New Guinea	MAG	AYMD	5.207080	145.789001	20	10.0	U	Pacific/Port_Moresby
2	3	Mount Hagen Kagamuga Airport	Mount Hagen	Papua New Guinea	HGU	AYMH	5.826790	144.296005	5388	10.0	U	Pacific/Port_Moresby
3	4	Nadzab Airport	Nadzab	Papua New Guinea	LAE	AYNZ	6.569803	146.725977	239	10.0	U	Pacific/Port_Moresby
4	5	Port Moresby Jacksons International Airport	Port Moresby	Papua New Guinea	POM	AYPY	9.443380	147.220001	146	10.0	U	Pacific/Port_Moresby

9) Explora los aeropuertos (I)

- ¿Países con más aeropuertos?

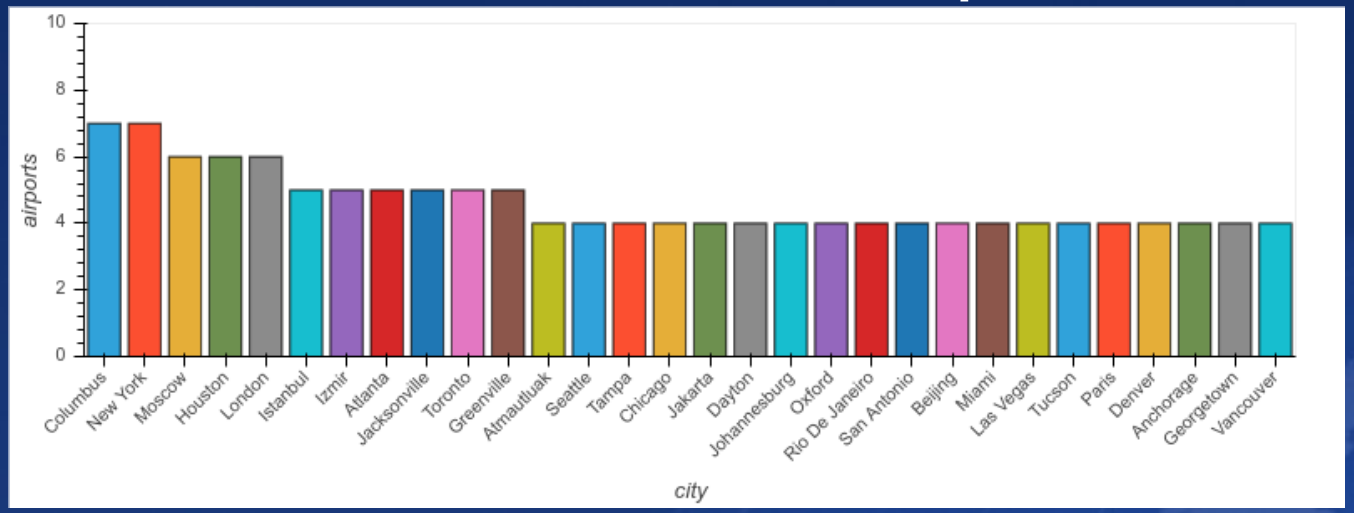


- ¿Aeropuertos de Madrid?

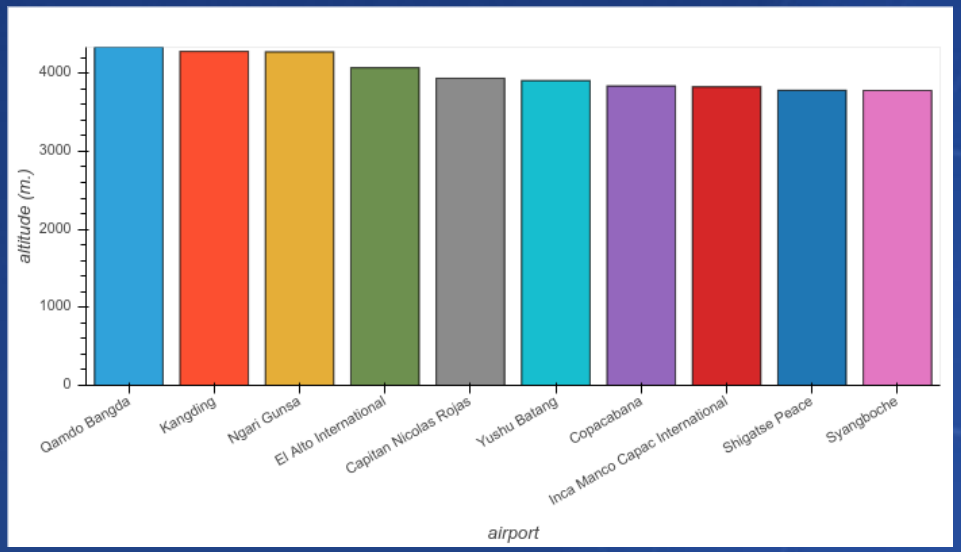
	Airport ID	Name	IATA	ICAO	Latitude	Longitude	Altitude	HoursFromUTC	DST	Timezone
1194	1224	Getafe Air Base	NaN	LEGT	40.294102	-3.72383	2031	1.0	E	Europe/Madrid
1198	1229	Adolfo Suárez Madrid–Barajas Airport	MAD	LEMD	40.471926	-3.56264	1998	1.0	E	Europe/Madrid
1214	1245	Torrejón Airport	TOJ	LETO	40.496700	-3.44587	2026	1.0	E	Europe/Madrid
10435	11878	Cuatro Vientos Airport	ECV	LECU	40.370701	-3.78514	2269	NaN	NaN	NaN

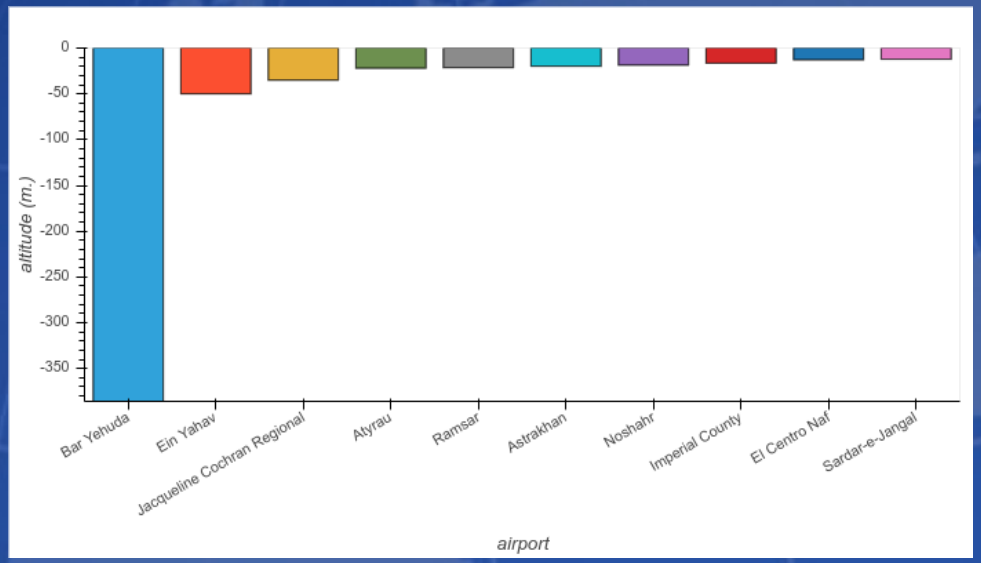
10) Explora los aeropuertos (II)

- ¿Ciudades con más aeropuertos?



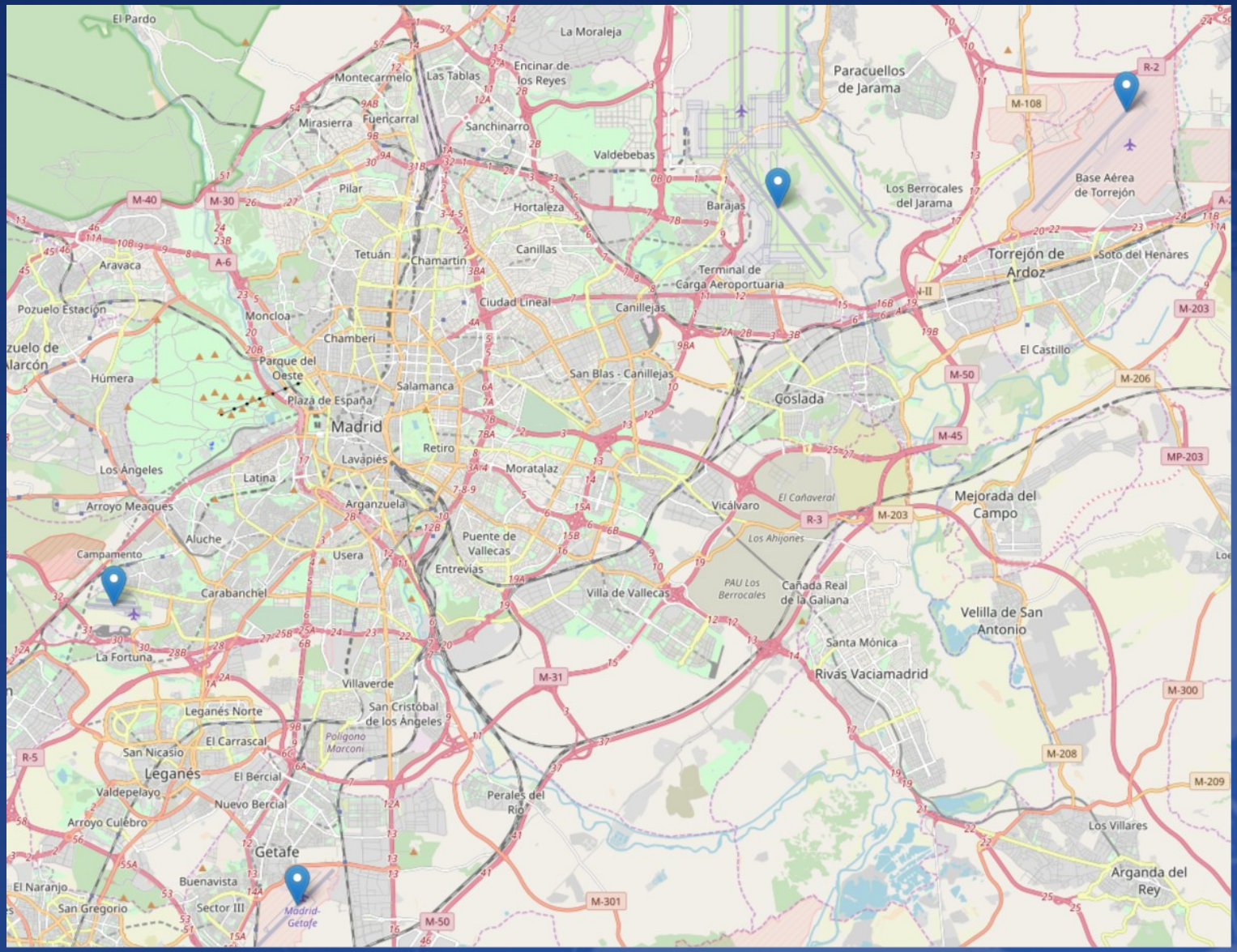
- ¿Aeropuertos a altitudes extremas?



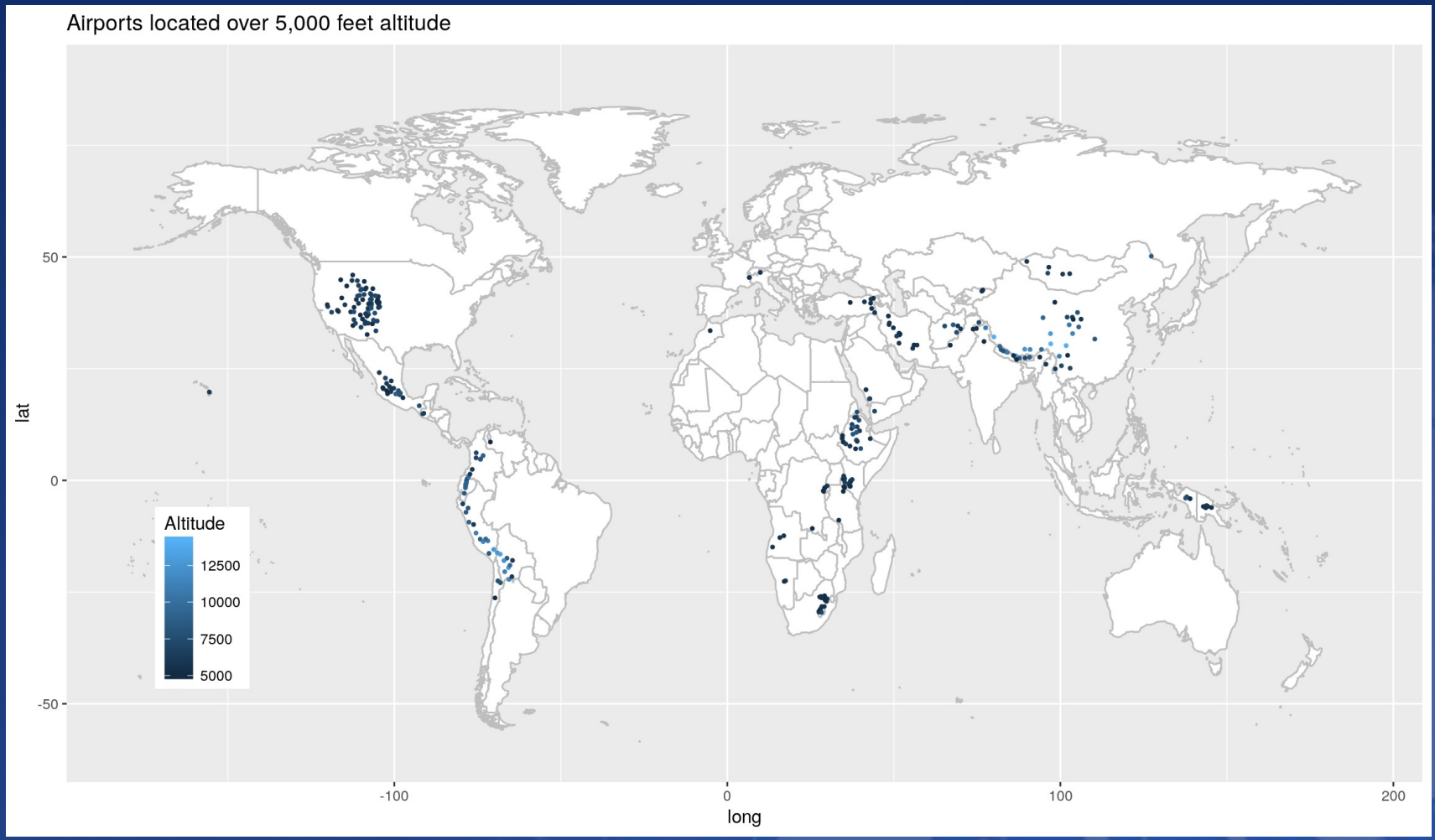


11) Explora los aeropuertos (III)

- ¿Localizaciones OK?



12) Explora los aeropuertos (IV)



13) Procesa y cruza las rutas

- 66000+ operativas.
- Todas con IDs de aerolínea y aeropuertos.

	airline	airline ID	source airport	source airport id	destination airport	destination airport id	codeshare	equipment
0	2B	410	AER	2965	KZN	2990	NaN	CR2
1	2B	410	ASF	2966	KZN	2990	NaN	CR2
2	2B	410	ASF	2966	MRV	2962	NaN	CR2
3	2B	410	CEK	2968	KZN	2990	NaN	CR2
4	2B	410	CEK	2968	OVB	4078	NaN	CR2

- Tras cruzar con ellos:

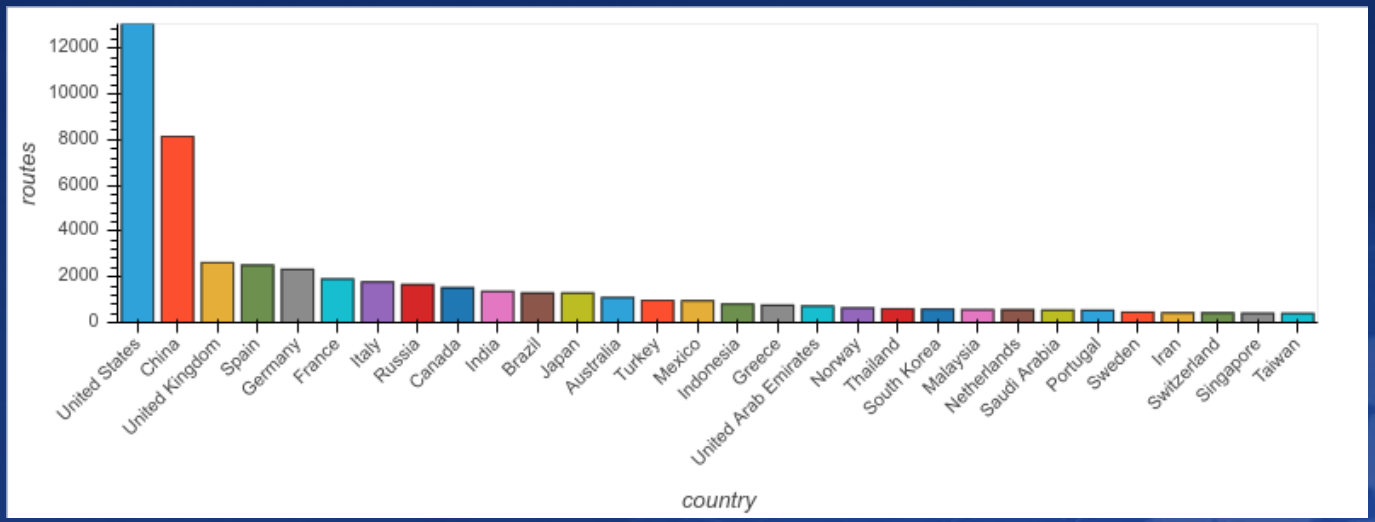
	airline name	destination name	destination city	destination country	destination latitude	destination longitude	source name	source city	source country	source latitude	source longitude
0	Aerocondor	Kazan International Airport	Kazan	Russia	55.606201	49.278702	Sochi International Airport	Sochi	Russia	43.449902	39.956600
1	Aerocondor	Kazan International Airport	Kazan	Russia	55.606201	49.278702	Astrakhan Airport	Astrakhan	Russia	46.283298	48.006302
2	Aerocondor	Kazan International Airport	Kazan	Russia	55.606201	49.278702	Chelyabinsk Balandino Airport	Chelyabinsk	Russia	55.305801	61.503300
3	Aerocondor	Kazan International Airport	Kazan	Russia	55.606201	49.278702	Domodedovo International Airport	Moscow	Russia	55.408798	37.906300
4	Aerocondor	Kazan International Airport	Kazan	Russia	55.606201	49.278702	Belgorod International Airport	Belgorod	Russia	50.643799	36.590099

14) Comenta el código hasta ahora

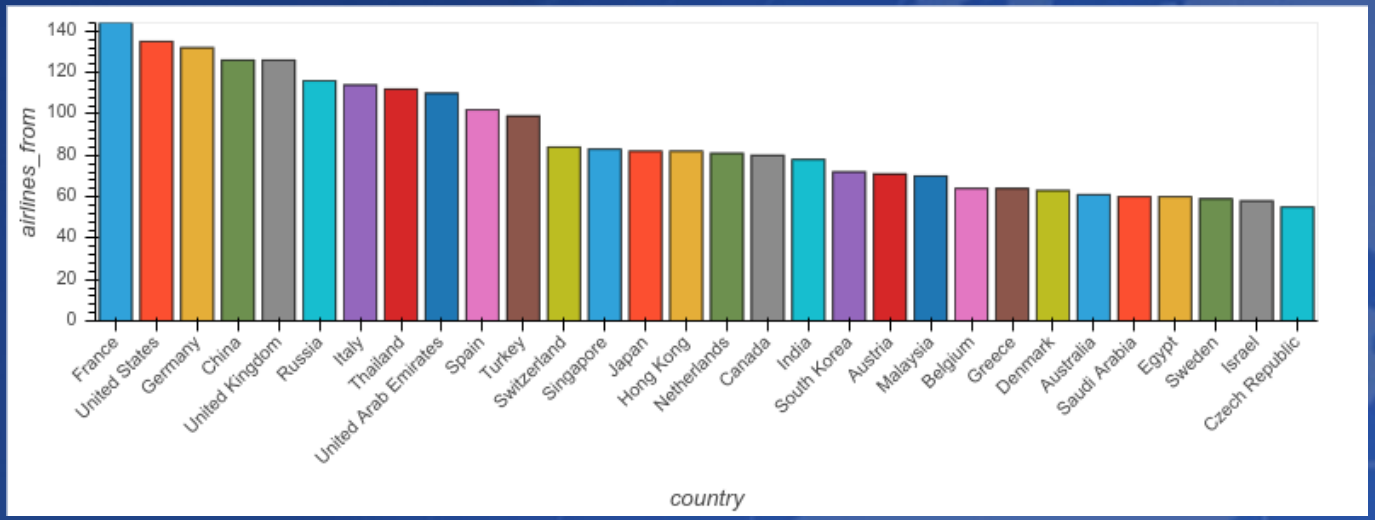
```
import pandas as pd
## Airlines
airlines = pd.read_csv('./data/airlines.csv', na_values='\\N')      # read airlines
airlines = airlines.loc[airlines['Active'] == 'Y', :].drop(columns='Active')      # remove non-active ones
## Airports, stations and ports
airports = pd.read_csv('./data/airports.csv', na_values='\\N', header=None)     # read airports
airports.rename(columns={0:'Airport ID', 1:'Name', 2:'City', 3:'Country', 4:'IATA', 5:'ICAO',
                6:'Latitude', 7:'Longitude', 8:'Altitude', 9:'HoursFromUTC', 10:'DST',
               11:'Timezone', 12:'Type', 13:'Source'}, inplace=True)    # manual renaming
airports = airports.loc[airports['Type'] == 'airport', :].drop(columns=['Type', 'Source'])  # just airports
## Routes
routes = pd.read_csv('./data/flights.csv', na_values='\\N')
cols = ['airline ID', 'source airport id', 'destination airport id']
routes = routes[routes['stops'] == 0].dropna(how='any', subset=cols).drop(columns='stops')  # direct routes
for c in cols:
    routes.loc[:, c] = routes[c].astype('int')   # no nans now, so cast to int
## Merge routes with all the rest
cols = ['Name', 'City', 'Country', 'Latitude', 'Longitude']
df = (pd.merge(routes, airports, left_on='source airport id', right_on='Airport ID', how='inner')
    .rename(columns={c:'source ' + c.lower() for c in cols})
    .drop(airports.columns, axis=1, errors='ignore')
    .merge(airports, left_on='destination airport id', right_on='Airport ID', how='inner')
    .rename(columns={c:'destination ' + c.lower() for c in cols})
    .drop(airports.columns, axis=1, errors='ignore')
    .merge(airlines, left_on='airline ID', right_on='Airline ID', how='inner')
    .rename(columns={'Name': 'airline name'})
    .drop(airlines.columns, axis=1, errors='ignore')
    .sort_index(axis=1)
)
```

15) Explora las rutas (I)

- ¿Países a los que llegan más rutas?

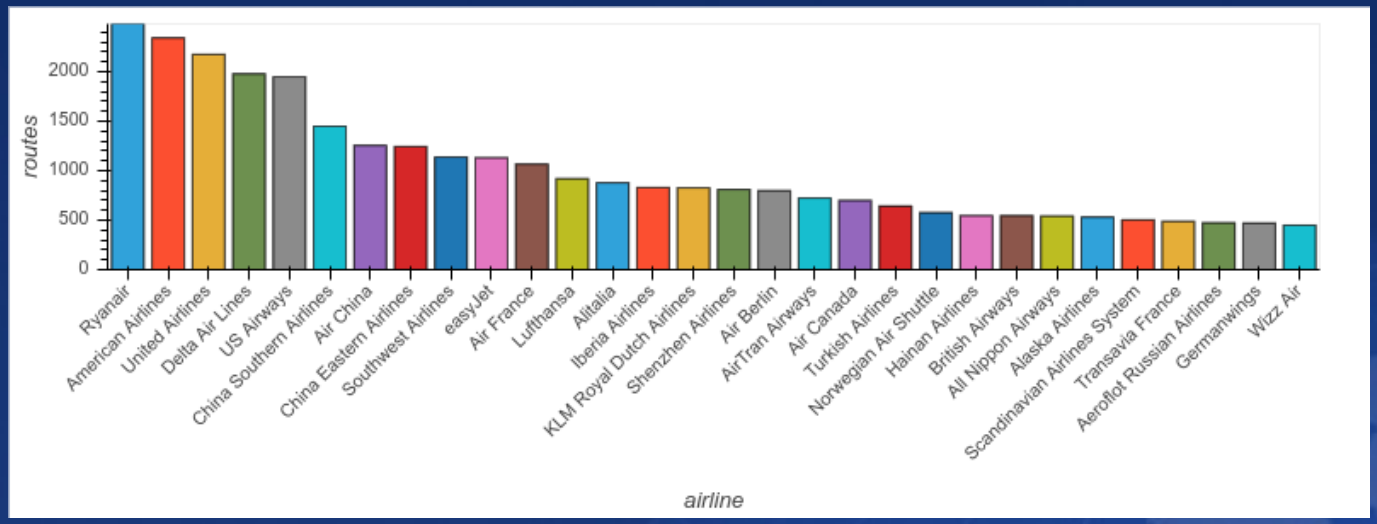


- ¿Países de los que salen más aerolíneas?

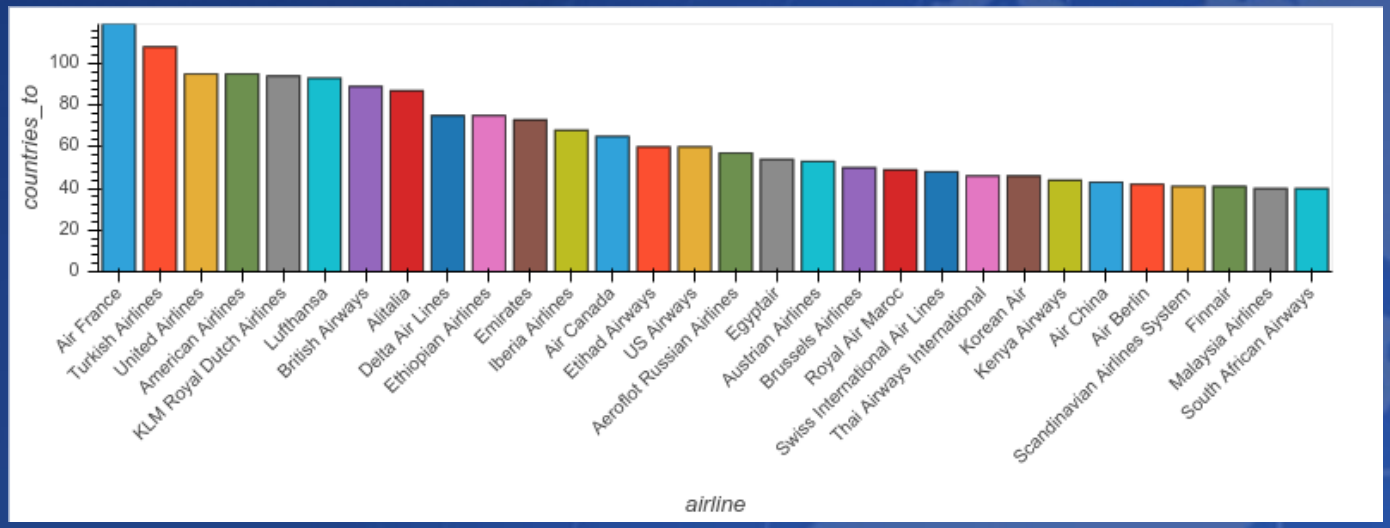


16) Explora las rutas (II)

- ¿Aerolíneas con más rutas?

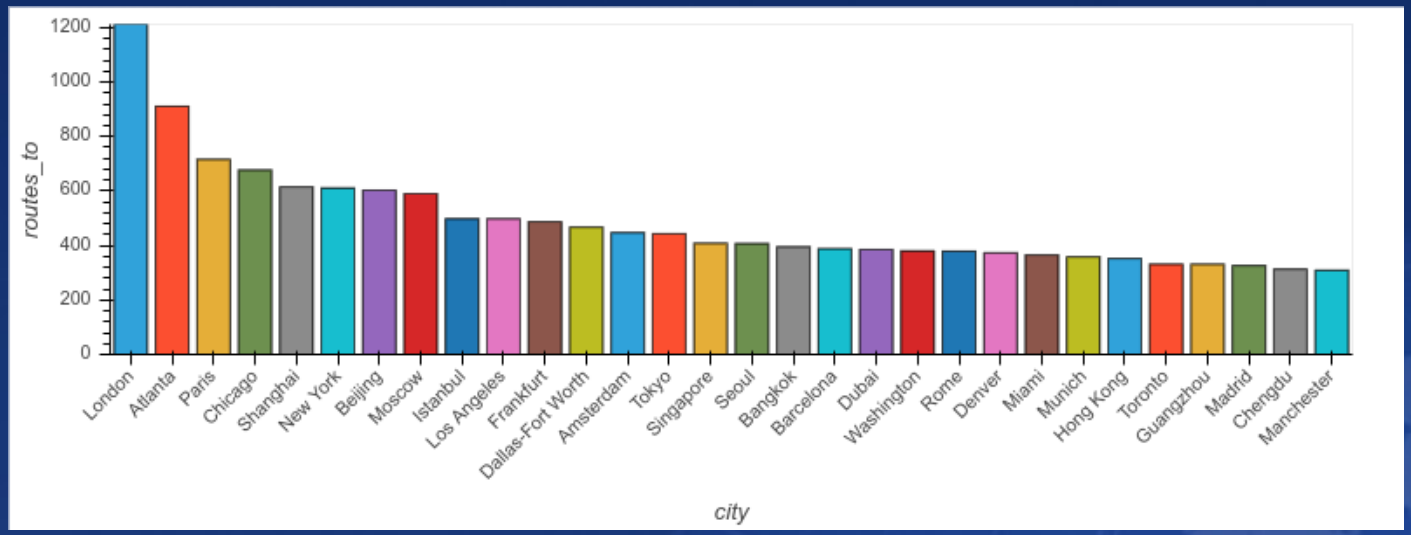


- ¿Aerolíneas que van a más países?

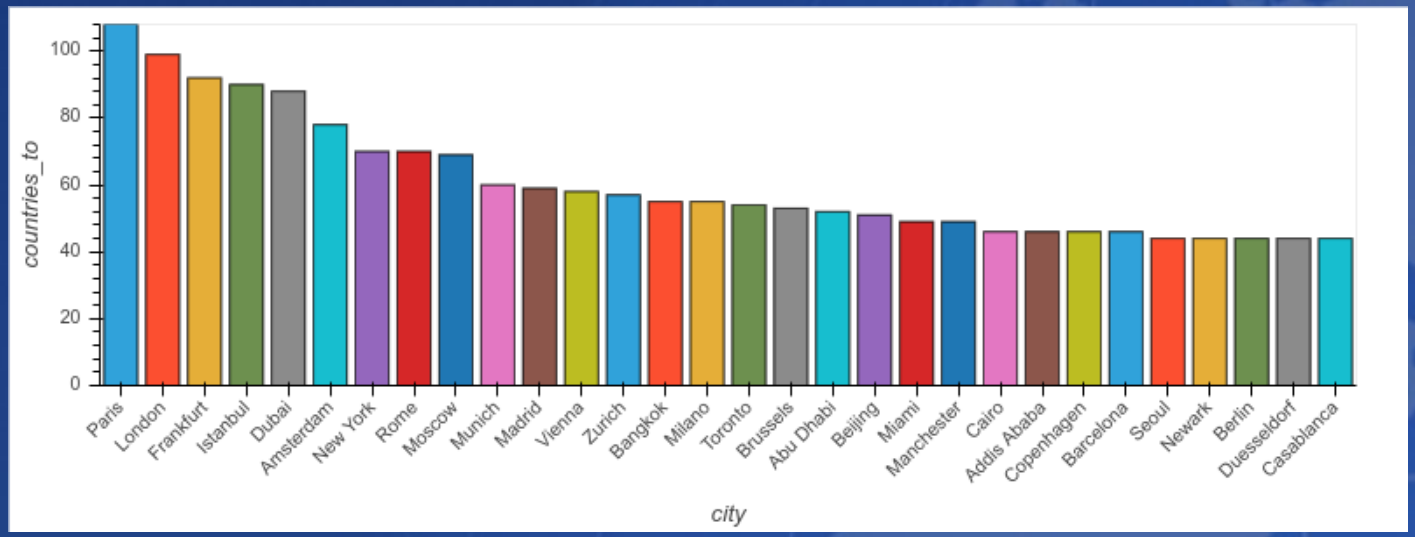


17) Explora las rutas (III)

- ¿Ciudades a las que van más rutas?

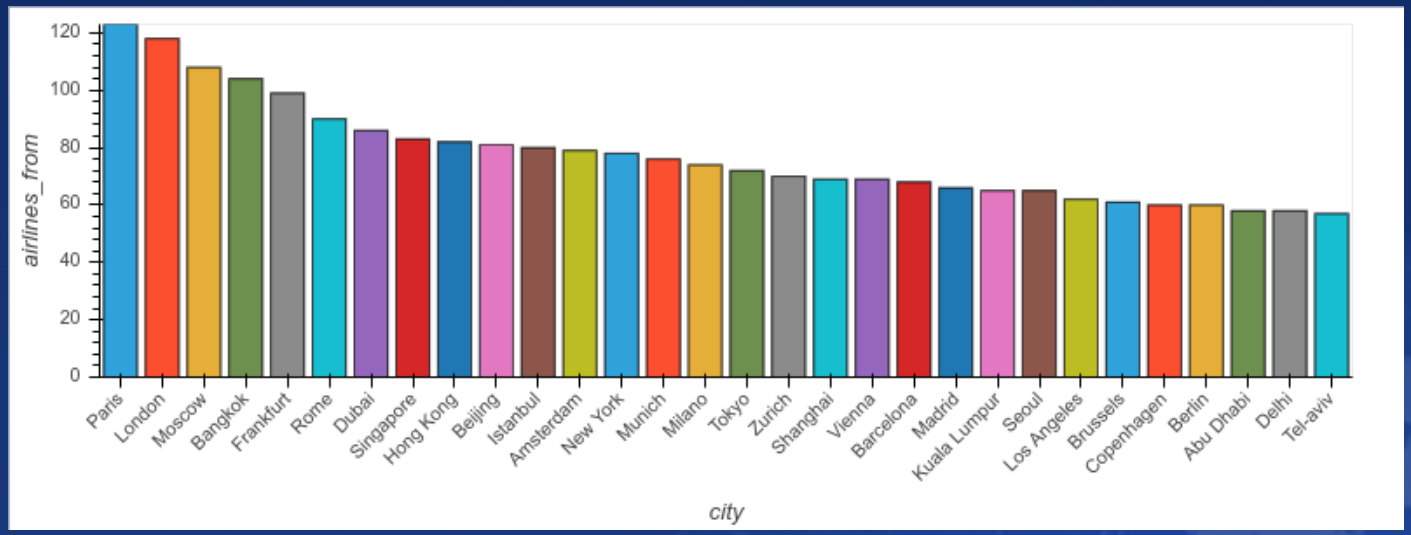


- ¿Ciudades desde las que se va a más países?

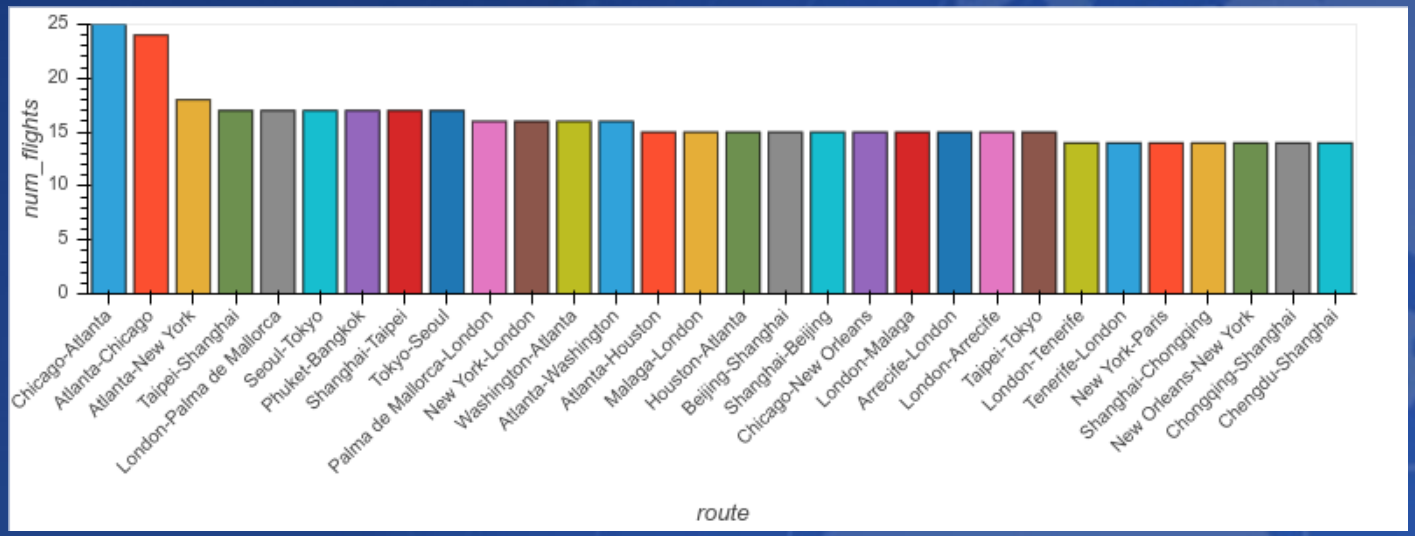


18) Explora las rutas (IV)

- ¿Ciudades con más aerolíneas?

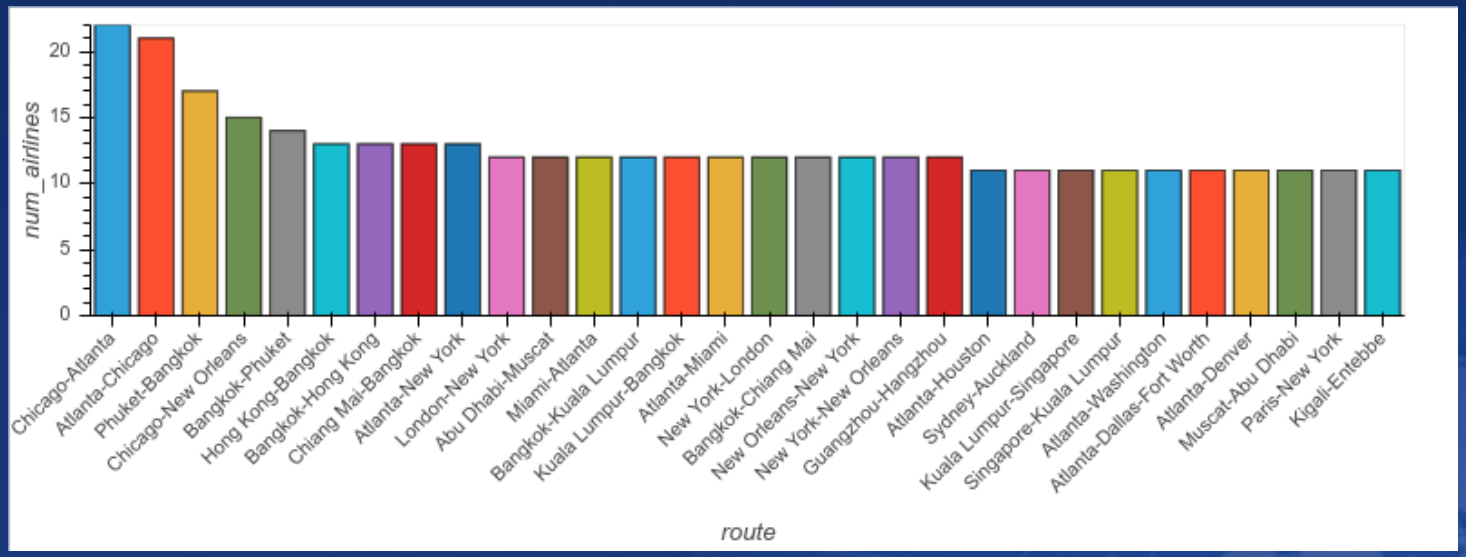


- ¿Rutas con más vuelos?



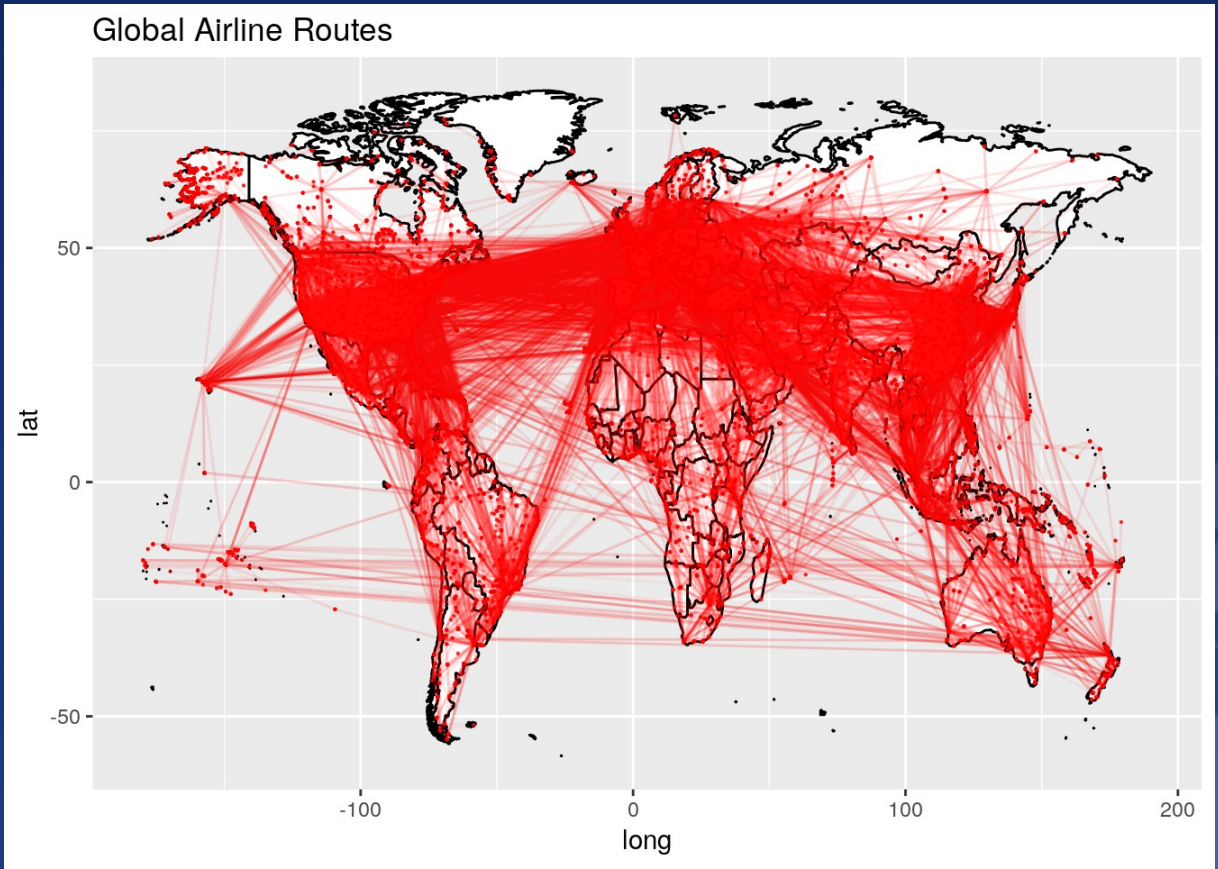
19) Explora las rutas (V)

- ¿Rutas que operan más aerolíneas?



- Y se podría seguir...
  - ¿Aeropuertos más al norte/sur/este/oeste?
  - ¿Aeropuertos más lejos de cualquier otro?
  - Pero eso ya implica calcular distancias y operar con coordenadas...

20) Intenta pintar las rutas tal cual



- Demasiadas, todo del mismo color y grosor, líneas rectas, no interactivo...
- ¡Y además hay un error gordo! ¿Cuál? ;-)

21) Piensa cómo pintarlas mejor

- Mantener aeropuertos con puntos y rutas con líneas → estructura de grafo.
- Interesa hacerlo por ciudad → filtro.
- Para ciudades populares, muchas rutas que salen de ahí → líneas finas y discontinuas.
- Al seleccionar una ruta mostrar de forma interactiva y dinámica:
  - Origen y destino (aeropuertos).
  - Aerolíneas que la operan.

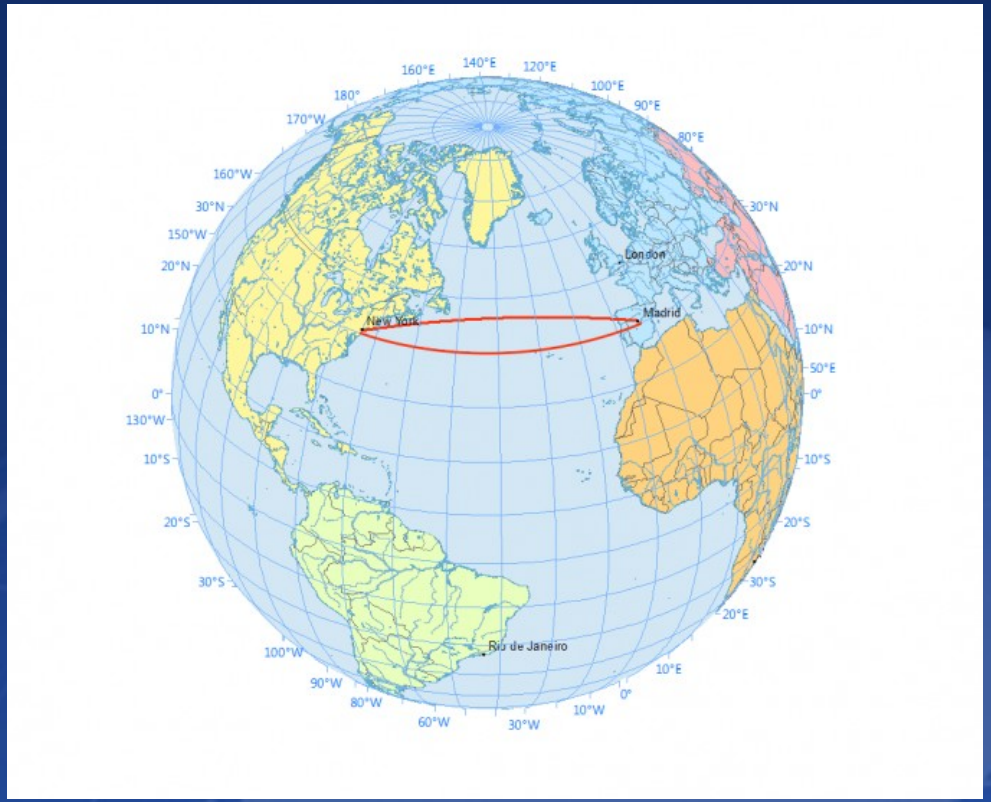
22) Pica el código que lo hace

```
import geoviews as gv ; gv.extension('bokeh')
def routes_from(city):
    # City routes
    depart_routes = df[(df['source city'] == city)].copy()
    depart_routes.loc[:, 'destination str'] = (depart_routes['destination name'].str.replace('Airport', '') + '('
                + depart_routes['destination airport'] + ')')
    depart_routes.loc[:, 'source str'] = (depart_routes['source name'].str.replace('Airport', '') + '('
                + depart_routes['source airport'] + ')')
    depart_routes.sort_index(axis=1, inplace=True)
    # Airlines per route
    aux = (depart_routes.groupby(['source str', 'destination str'])
       .agg({'airline name': 'unique',
             'source latitude': 'first', 'source longitude': 'first', 'source city': 'first',
             'destination latitude': 'first', 'destination longitude': 'first', 'destination city': 'first'}))
    aux.rename(columns={'airline name': 'airlines'}, inplace=True)
    aux.loc[:, 'num_airlines'] = aux['airlines'].apply(len)
    aux.loc[:, 'airlines'] = aux['airlines'].str.join(', ')    # from list to string
    aux.reset_index(inplace=True)
    # Build nodes with airports involved
    dest_airports = aux.filter(regex='destination *').drop_duplicates()
    src_airports = aux.filter(regex='source *').drop_duplicates()
    src_airports.columns = dest_airports.columns     # as if they were also destinations
    locs = pd.concat([src_airports, dest_airports], axis=0)
    nodes = gv.Nodes(locs, kdims=[('destination longitude', 'lon'), ('destination latitude', 'lat'),
                                  ('destination str', 'airport')], vdims=[('destination city', 'city')])
    # Build graph
    graph = gv.Graph((aux[['source str', 'destination str', 'num_airlines', 'airlines']], nodes, None),
         kdims=[('source str', 'from'), ('destination str', 'to')], vdims=['airlines'])
    return graph
```

23) Prueba a ver qué tal funciona

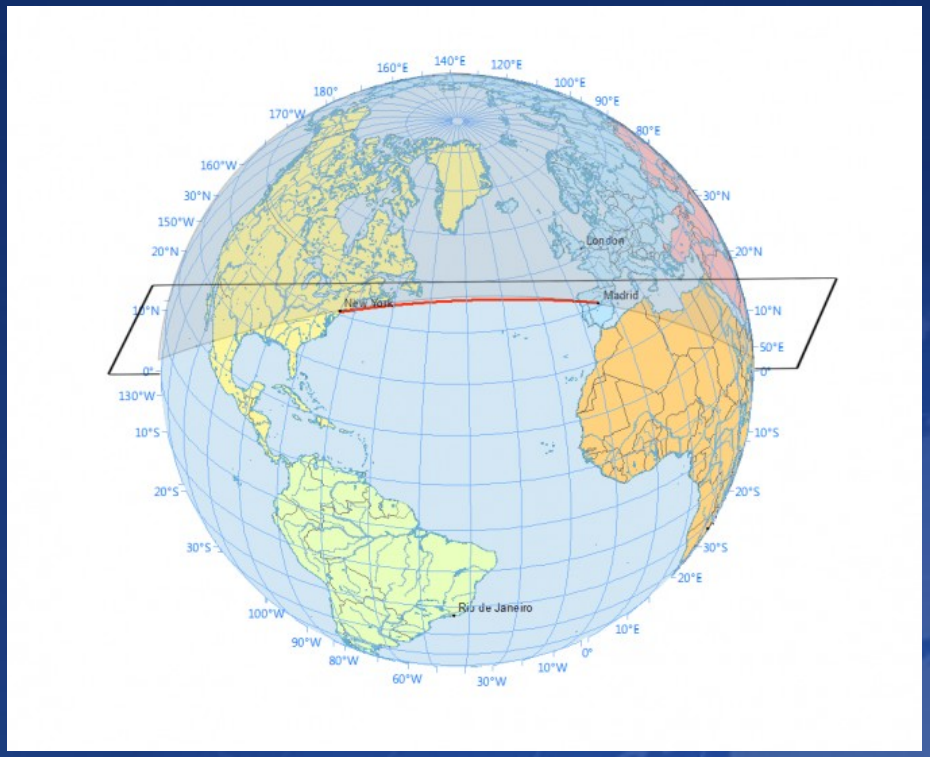
- Bastante aparente, aeropuertos OK...
- ¡Pero las rutas realmente no son así!
- Por ejemplo, entre Madrid y Nueva York...

24) Descubre el mundo de la geodesia (I)



- ¡Camino más corto no es en línea recta!
- Distorsión al proyectar de 3D a 2D hace parecer que es la recta, ¡pero no!

25) Descubre el mundo de la geodesia (II)



- Es el corte que pasa por las 2 ciudades y divide la Tierra en 2 mitades iguales.
- Equivalente a meridiano que pase por ambas.

26) Busca qué librerías implementan cálculos geodésicos

- La precursora: GeographicLib.
- Librería general de proyecciones y cambios de coordenadas: PROJ.4.
- Versión Python: pyproj.
- Cálculos precisos con elipsoides (¡en realidad la Tierra no es una esfera!).
- ¡Función npts es justo lo que queremos! (dados origen y destino, devuelve puntos a lo largo de la línea geodésica).

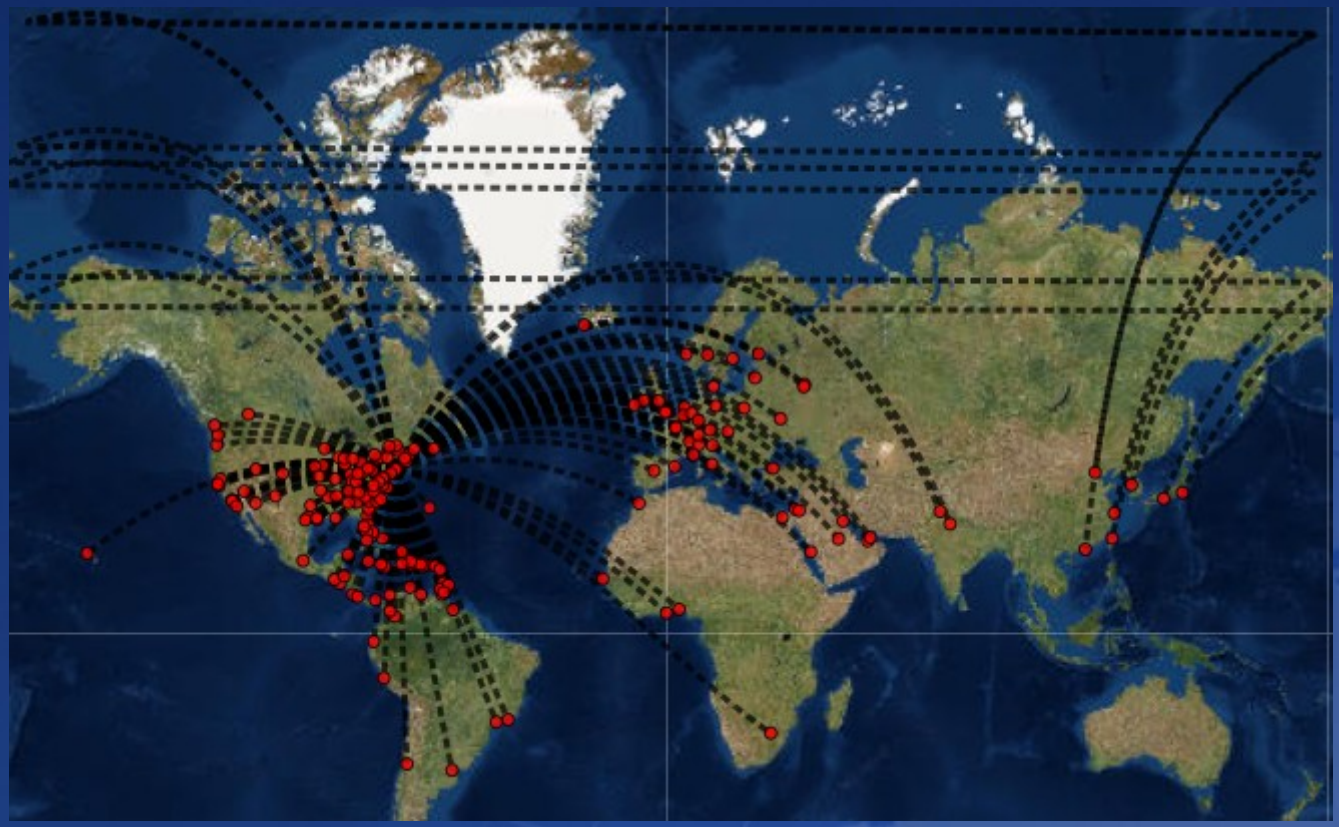
27) Modifica el código para que calcule líneas geodésicas

```
from pyproj import Geod
import numpy as np
from shapely.geometry import LineString
def routes_from(city, geodesic=False):
    ####### Same steps as before to get nodes #######
    # Edges and paths
    if geodesic:
        # Build edges with geodesic curves, instead of straight lines
        edges = (aux
         .apply(lambda row: LineString(np.array([(row['source longitude'], row['source latitude'])] +
                Geod(ellps='clrk66').npts(row['source longitude'], row['source latitude'],
                       row['destination longitude'], row['destination latitude'], npts=100) +  # 100 points
                [(row['destination longitude'], row['destination latitude'])])), axis=1))
        edgepaths = gv.EdgePaths(gpd.GeoDataFrame(geometry=edges),
                         kdims=[('destination longitude', 'lon'), ('destination latitude', 'lat')])
    else:
        # Build edges right away
        edgepaths = None
    # Build graph
    graph = gv.Graph((aux[['source str', 'destination str', 'num_airlines', 'airlines']], nodes, edgepaths),
         kdims=[('source str', 'from'), ('destination str', 'to')], vdims=['airlines'])
    return graph
```

28) Vuelve a ver qué tal funciona (I)

- ¡Madrid ahora genial!
- Pero si se prueba con Nueva York...

29) Vuelve a ver qué tal funciona (II)



- Problema con vuelos que tienen que pasar de un lado al otro del mapa...
- ¡Esas líneas hay que partirlas en 2!

30) Vuelve a modificar el código para solucionar el problema

```
def repair_line(l):
    coords = list(l._get_coords())
    res = [] ; prev_lon = 0.0 ; prev_lat = 0.0 ; ini = 0
    for pos, (lon, lat) in enumerate(coords):
        if (prev_lon < -150.0 and lon > 150.0) or (prev_lon > 150.0 and lon < -150.0):
            res.append(coords[ini:pos]) ; ini = pos
        prev_lon, prev_lat = lon, lat
    res.append(coords[ini:pos])
    return MultiLineString(res) if len(res) > 1 else l
```

```
# Edges and paths
if geodesic:
    # Build edges with geodesic curves, instead of straight lines
    edges = (aux
     .apply(lambda row: LineString(np.array([(row['source longitude'], row['source latitude'])] +
            Geod(ellps='clrk66').npts(row['source longitude'], row['source latitude'],
                   row['destination longitude'], row['destination latitude'], npts=100) +    # 100 points
            [(row['destination longitude'], row['destination latitude'])])), axis=1))
    edgepaths = gv.EdgePaths(gpd.GeoDataFrame(geometry=edges.apply(repair_line)),
                                          kdims=[('destination longitude', 'lon'),
                                                 ('destination latitude', 'lat')])
else:
    # Build edges right away
    edgepaths = None
# Build graph
graph = gv.Graph((aux[['source str', 'destination str', 'num_airlines', 'airlines']], nodes, edgepaths),
     kdims=[('source str', 'from'), ('destination str', 'to')], vdims=['airlines'])
return graph
```

31) Comprueba que ahora todo funciona

- Vale, pero falta comprobar la interactividad...

32) Muestra una demo del resultado y enlaza el código final

- En Github:
  - Código (notebook).
  - Datos (CSV).
- En Medium:
  - Artículo de divulgación (en inglés).

33) Agradece la asistencia tal día como hoy