In [1]:
```
import pandas as pd
```

In [2]:
```
languages = pd.read_csv('languages_and_dialects_geo.csv')
```

In [3]:
```
print(languages.describe(include='all'))
```

```
       glottocode        name isocodes    level macroarea     latitude  \
count       22111       22111     8121    22111     22019  8849.000000
unique      22111       22111     8121        2         6          NaN
top      3adt1234  3Ad-Tekles      aiw  dialect   Eurasia          NaN
freq            1           1        1    13507      7011          NaN
mean          NaN         NaN      NaN      NaN       NaN     8.935550
std           NaN         NaN      NaN      NaN       NaN    19.603034
min           NaN         NaN      NaN      NaN       NaN   -55.274800
25%           NaN         NaN      NaN      NaN       NaN    -5.086930
50%           NaN         NaN      NaN      NaN       NaN     6.842170
75%           NaN         NaN      NaN      NaN       NaN    21.572100
max           NaN         NaN      NaN      NaN       NaN    73.135400

          longitude
count   8849.000000
unique          NaN
top             NaN
freq            NaN
mean      50.715108
std       80.979511
min     -178.785000
25%        7.213910
50%       47.837900
75%      123.156317
max      179.306000
```

In [4]:
```
print(f"The database has {len(languages)} entries on languiges and databa
```

```
The database has 22111 entries on languiges and databases.
```

In [5]:
```
print(f"The full area of macroareas are{languages['macroarea'].unique()}"
```

```
The full area of macroareas are['Africa' 'Papunesia' 'Eurasia' 'South Ame
rica' 'North America'
 'Australia' nan]
```

In [6]:
```
print(f"There are {languages['isocodes'].notna().sum()} languages and dia
```

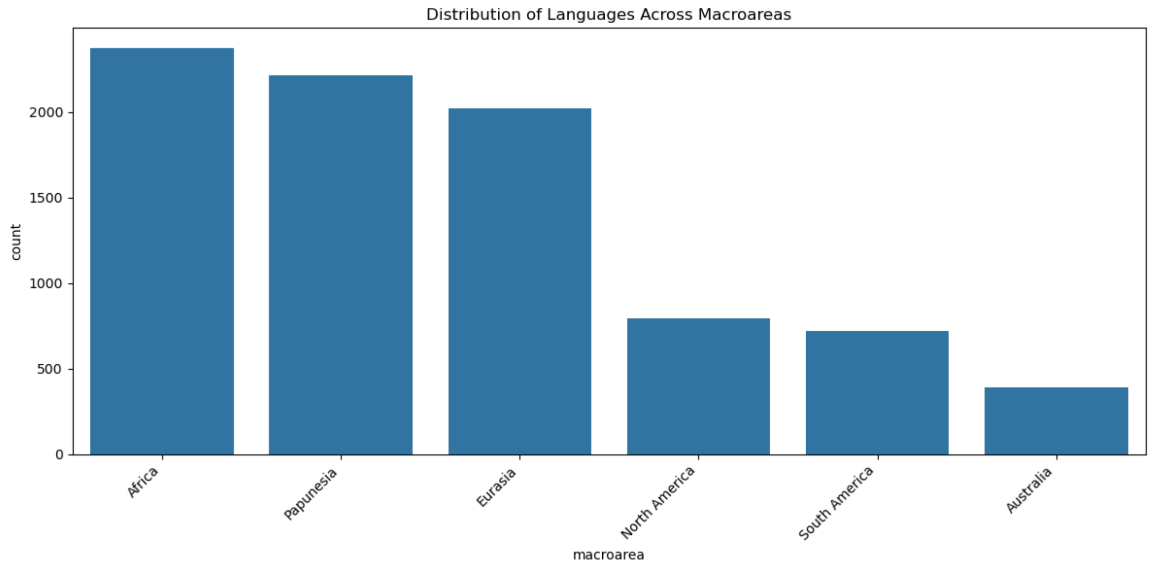
```
There are 8121 languages and dialects with the ISO 639-3 code.
```

In [7]:
```
print(f"We have latitude and longitude data for {languages['latitude'].no
```

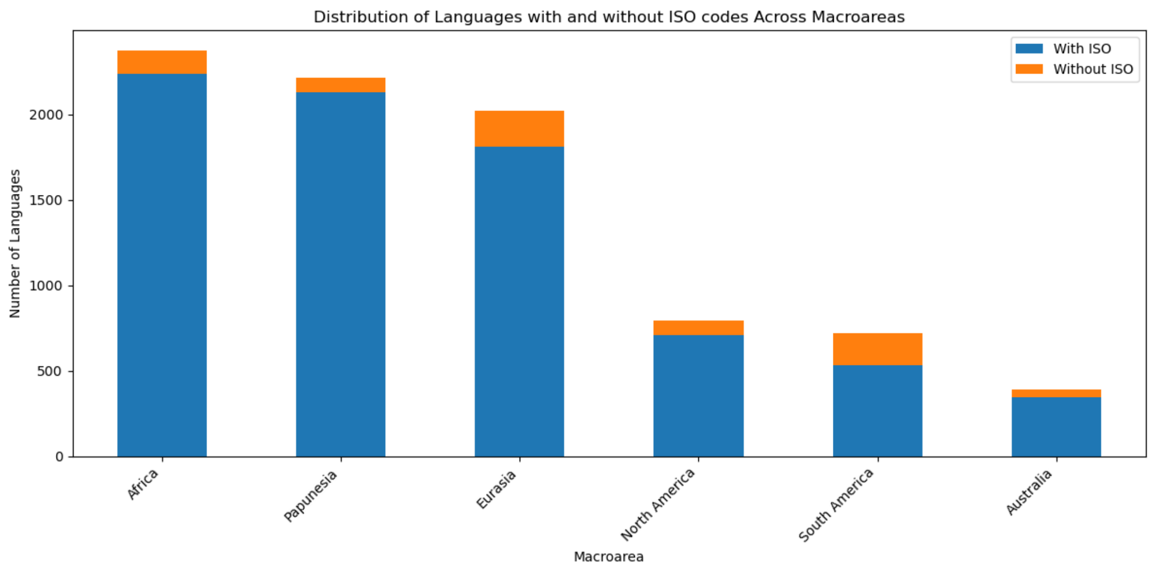
```
We have latitude and longitude data for 8849 languages and dialects.
```

In [8]:
```
languages = languages[languages['level'] != 'dialect']
```

In [9]:
```
import seaborn as sns

import matplotlib.pyplot as plt

plt.figure(figsize=(12, 6))
sns.countplot(data=languages, x='macroarea', order = languages['macroarea
plt.title('Distribution of Languages Across Macroareas')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



In [10]:
```
macroarea_counts = languages['macroarea'].value_counts()
macroarea_iso_counts = languages[languages['isocodes'].notna()]['macroare
df_macroarea = pd.DataFrame({'Total': macroarea_counts, 'With ISO': macro

df_macroarea['Without ISO'] = df_macroarea['Total'] - df_macroarea['With
df_macroarea[['With ISO', 'Without ISO']].plot(kind='bar', stacked=True,
plt.title('Distribution of Languages with and without ISO codes Across Ma
plt.xlabel('Macroarea')
plt.ylabel('Number of Languages')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```



In [11]:
```
northernmost = languages.loc[languages['latitude'].idxmax()]
southernmost = languages.loc[languages['latitude'].idxmin()]

print("Northernmost Language:")
print(f"Name: {northernmost['name']}, Macroarea: {northernmost['macroarea

print("\nSouthernmost Language:")
print(f"Name: {southernmost['name']}, Macroarea: {southernmost['macroarea
```

```
Northernmost Language:
Name: Nganasan, Macroarea: Eurasia, Latitude: 73.1354

Southernmost Language:
Name: Yámana, Macroarea: South America, Latitude: -55.2748
```

In [12]:
```
tropical_languages = languages[(languages['latitude'] >= -23.43619) & (la
percentage = (len(tropical_languages) / len(languages)) * 100

print(f"{percentage:.2f}% of the world's languages are spoken in the trop
```

```
73.18% of the world's languages are spoken in the tropics.
```

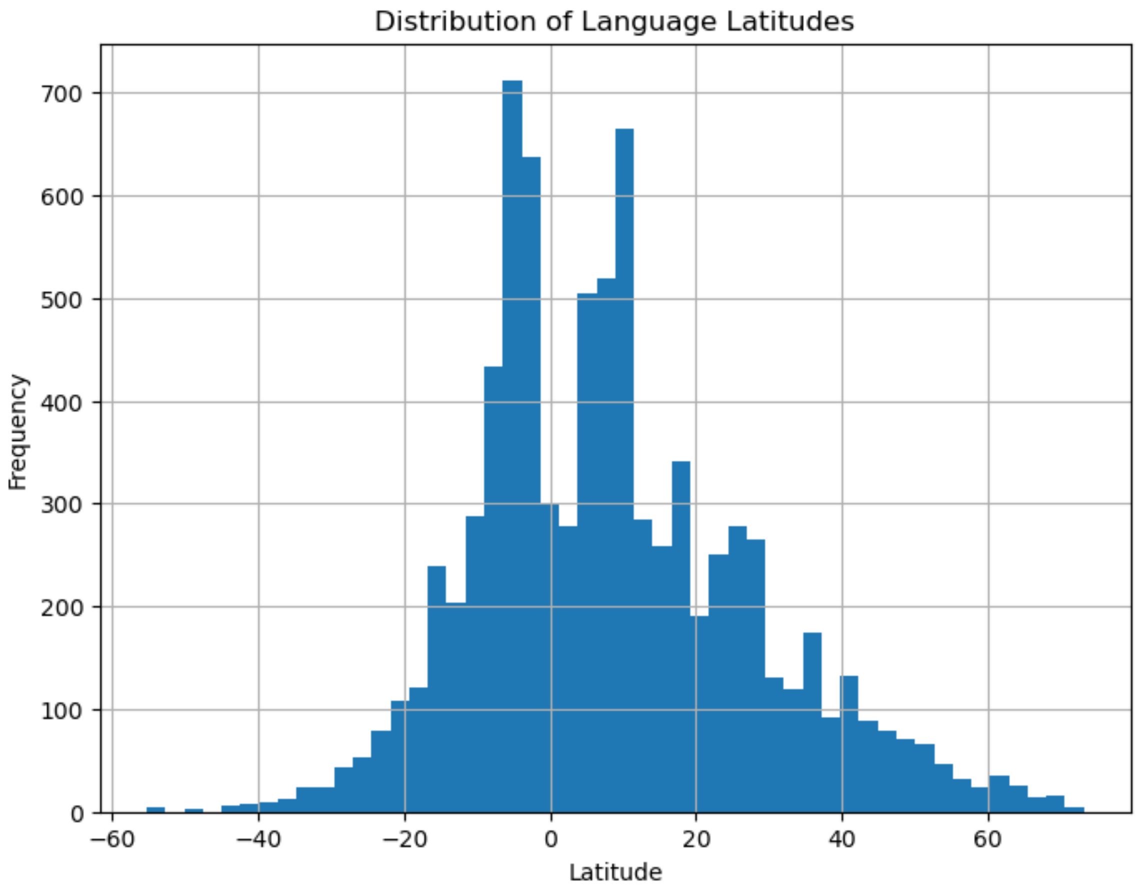
In [13]:
```
#3c

import numpy as np

def haversine(lat1, lon1, lat2, lon2):
    R = 6371  # Radius of Earth in kilometers
    lat1, lon1, lat2, lon2 = map(np.radians, [lat1, lon1, lat2, lon2])

    a = np.sin((lat2 - lat1)/2)**2 + np.cos(lat1) * np.cos(lat2) * np.sin
    c = 2 * np.arcsin(np.sqrt(a))
    return R * c

languages['distance_to_northernmost'] = languages.apply(
    lambda row: haversine(northernmost['latitude'], northernmost['longitu
    axis=1
)

furthest_language = languages.loc[languages['distance_to_northernmost'].i

print("Language Furthest from Northernmost Language:")
print(f"Name: {furthest_language['name']}, Macroarea: {furthest_language[
print(f"Distance: {furthest_language['distance_to_northernmost']} km")

print("\nSouthernmost Language:")
print(f"Name: {southernmost['name']}, Macroarea: {southernmost['macroarea

distance_southernmost_to_northernmost = haversine(northernmost['latitude'
print(f"\nDistance between Northernmost and Southernmost Language: {dista

if furthest_language['name'] == southernmost['name']:
    print("\nThe language furthest from the northernmost language is the
else:
    print("\nThe language furthest from the northernmost language is NOT
```

```
Language Furthest from Northernmost Language:
Name: Yámana, Macroarea: South America, Latitude: -55.2748, Longitude: -6
8.2648
Distance: 17717.541945364123 km

Southernmost Language:
Name: Yámana, Macroarea: South America, Latitude: -55.2748, Longitude: -6
8.2648

Distance between Northernmost and Southernmost Language: 17717.5419453641
23 km

The language furthest from the northernmost language is the southernmost
language.
```

In [14]:

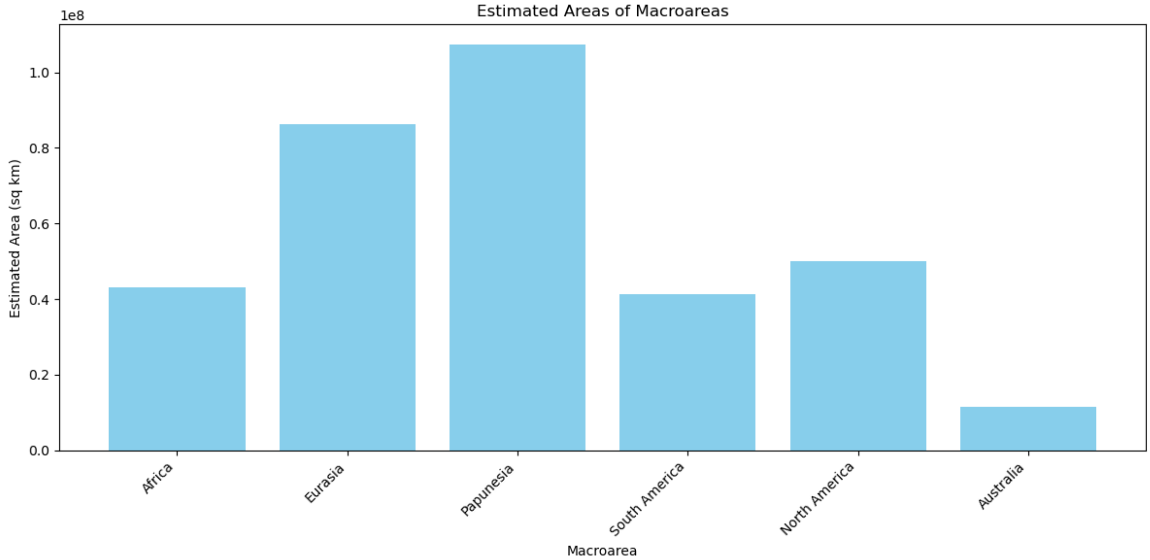
```
q1 = languages['latitude'].quantile(0.25)
q3 = languages['latitude'].quantile(0.75)

print(f"Q1 (25th percentile): {q1}")
print(f"Q3 (75th percentile): {q3}")

if abs(q1 + q3) < 1e-6:
    print("The latitude range is symmetric around the equator.")
else:
    print("The latitude range is not symmetric around the equator.")

import matplotlib.pyplot as plt

plt.figure(figsize=(8, 6))
languages['latitude'].hist(bins=50)
plt.title('Distribution of Language Latitudes')
plt.xlabel('Latitude')
plt.ylabel('Frequency')
plt.show()
```

```
Q1 (25th percentile): -5.0245
Q3 (75th percentile): 20.16
The latitude range is not symmetric around the equator.
```


Distribution of Language Latitudes
Frequency
Latitude
0
100
200
300
400
500
600
700
−60
−40
−20
0
20
40
60

In [15]:
```
import numpy as np

def estimate_area(macroarea_name, num_pairs=1000):
    macroarea_languages = languages[languages['macroarea'] == macroarea_n

    if len(macroarea_languages) < 2:
        print(f"Not enough language data for {macroarea_name} to estimate
        return None

    distances = []
    for _ in range(num_pairs):
        if len(macroarea_languages) < 2:
            break
        lang_sample = macroarea_languages.sample(2)
        lang1 = lang_sample.iloc[0]
        lang2 = lang_sample.iloc[1]
        dist = haversine(lang1['latitude'], lang1['longitude'], lang2['la
        distances.append(dist)

    if not distances:
        print(f"Could not compute distances for {macroarea_name}.")
        return None

    distances = sorted(distances, reverse=True)
    a = distances[0]
    b = distances[1] if len(distances) > 1 else a  # If only one distance

    area = np.pi * (a / 2) * (b / 2)
    return area

macroareas = languages['macroarea'].unique()
macroarea_areas = {}

for macroarea in macroareas:
    if isinstance(macroarea, str):
        area = estimate_area(macroarea)
        if area is not None:
            macroarea_areas[macroarea] = area

print("Estimated Areas of Macroareas:")
for macroarea, area in macroarea_areas.items():
    print(f"{macroarea}: {area:.2f} sq km")


macroarea_names = list(macroarea_areas.keys())
areas = list(macroarea_areas.values())

plt.figure(figsize=(12, 6))
plt.bar(macroarea_names, areas, color='skyblue')
plt.xlabel('Macroarea')
plt.ylabel('Estimated Area (sq km)')
plt.title('Estimated Areas of Macroareas')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
Estimated Areas of Macroareas:
Africa: 43124104.65 sq km
Eurasia: 86260224.11 sq km
Papunesia: 107304827.27 sq km
South America: 41252223.78 sq km
North America: 50065909.26 sq km
Australia: 11372286.29 sq km
```

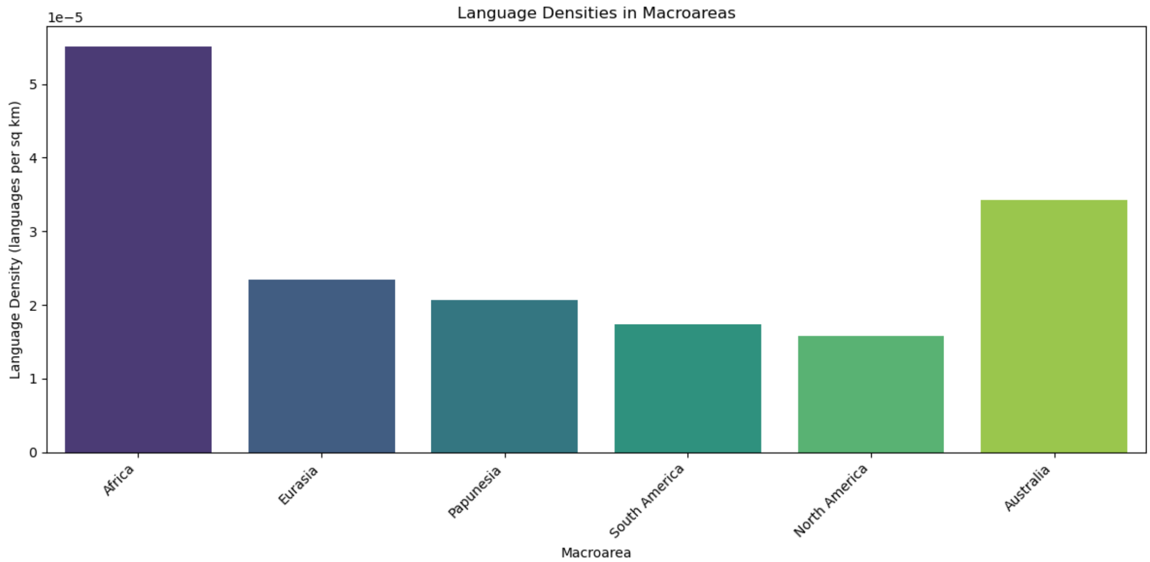


In [16]:

```python
macroarea_densities = {}

for macroarea in macroareas:
    if isinstance(macroarea, str) and macroarea in macroarea_areas:
        num_languages = len(languages[languages['macroarea'] == macroarea
        area = macroarea_areas[macroarea]
        density = num_languages / area
        macroarea_densities[macroarea] = density

print("Language Densities in Macroareas:")
for macroarea, density in macroarea_densities.items():
    print(f"{macroarea}: {density:.2f} languages per sq km")

macroarea_names = list(macroarea_densities.keys())
densities = list(macroarea_densities.values())

plt.figure(figsize=(12, 6))
sns.barplot(x=macroarea_names, y=densities, palette="viridis")
plt.xlabel('Macroarea')
plt.ylabel('Language Density (languages per sq km)')
plt.title('Language Densities in Macroareas')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()
```

```
Language Densities in Macroareas:
Africa: 0.00 languages per sq km
Eurasia: 0.00 languages per sq km
Papunesia: 0.00 languages per sq km
South America: 0.00 languages per sq km
North America: 0.00 languages per sq km
Australia: 0.00 languages per sq km
```

```
/tmp/ipykernel_8770/555148234.py:18: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be remov
ed in v0.14.0. Assign the `x` variable to `hue` and set `legend=False` fo
r the same effect.

  sns.barplot(x=macroarea_names, y=densities, palette="viridis")
```



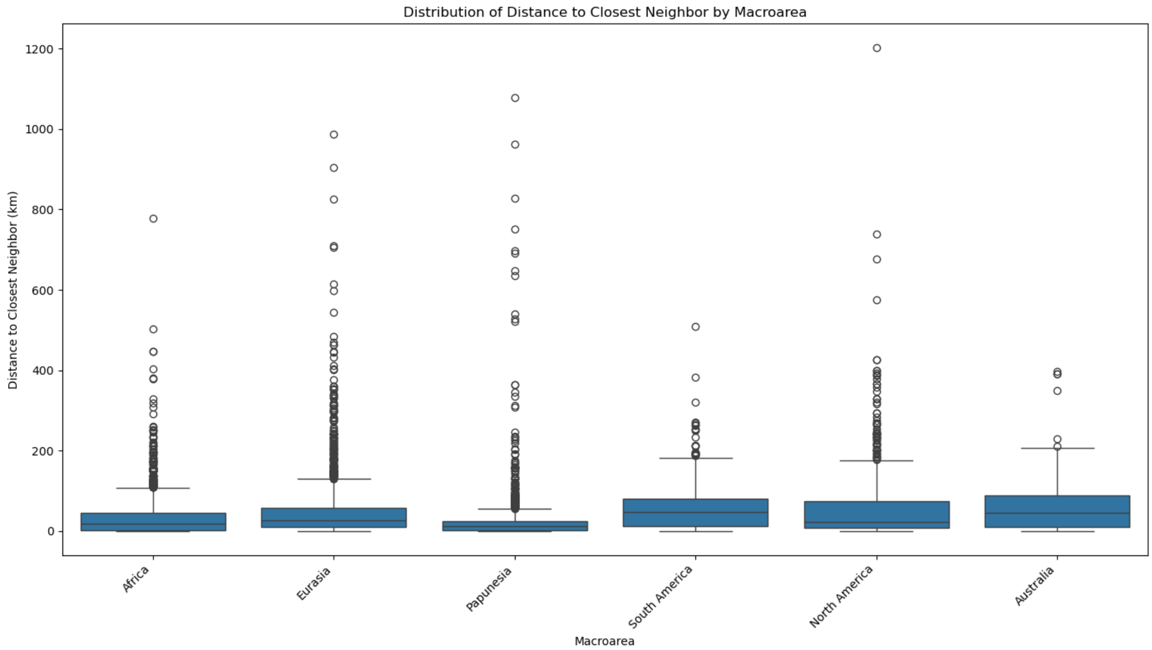
In [17]:

```
languages_sorted = languages.dropna(subset=['longitude']).sort_values('lo

max_longitude_diff = 0
language1 = None
language2 = None

for i in range(len(languages_sorted) - 1):
    diff = languages_sorted['longitude'].iloc[i+1] - languages_sorted['lo
    if diff > max_longitude_diff:
        max_longitude_diff = diff
        language1 = languages_sorted['name'].iloc[i]
        language2 = languages_sorted['name'].iloc[i+1]

print(f"Largest gap in longitude: {max_longitude_diff:.2f} degrees betwee
```

```
Largest gap in longitude: 12.54 degrees between Tarairiu and Kabuverdianu
```

In [18]:
```
# Filter out dialects to speed up processing
languages_filtered = languages[languages['level'] == 'language']
print("Checkpoint: Filtered out dialects. Remaining languages:", len(lang

# Define the vectorized haversine function
def haversine_vectorized(lat1, lon1, lat2, lon2):
    R = 6371  # Radius of Earth in kilometers
    lat1, lon1, lat2, lon2 = map(np.radians, [lat1, lon1, lat2, lon2])
    return np.arccos(np.sin(lat1) * np.sin(lat2) + np.cos(lat1) * np.cos(

# Compute the distance to the closest neighbor for each language
def compute_closest_distance(language, all_languages):
    lat1, lon1 = language['latitude'], language['longitude']
    distances = haversine_vectorized(lat1, lon1, all_languages['latitude'
    distances = distances[distances > 0]  # Exclude self-distance (0)
    return distances.min() if len(distances) > 0 else None

print("Checkpoint: Starting to compute distances to closest neighbors.")
languages_filtered['distance_to_closest'] = languages_filtered.apply(
    lambda row: compute_closest_distance(row, languages_filtered), axis=1
)
print("Checkpoint: Finished computing distances to closest neighbors.")

# Plot the distributions as a boxplot
plt.figure(figsize=(14, 8))
sns.boxplot(x='macroarea', y='distance_to_closest', data=languages_filter
plt.title('Distribution of Distance to Closest Neighbor by Macroarea')
plt.xlabel('Macroarea')
plt.ylabel('Distance to Closest Neighbor (km)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

# Identify the most isolated languages
isolated_languages = languages_filtered.sort_values(by='distance_to_close
print("\nTop 10 Most Isolated Languages:")
print(isolated_languages[['name', 'macroarea', 'distance_to_closest', 'la
```

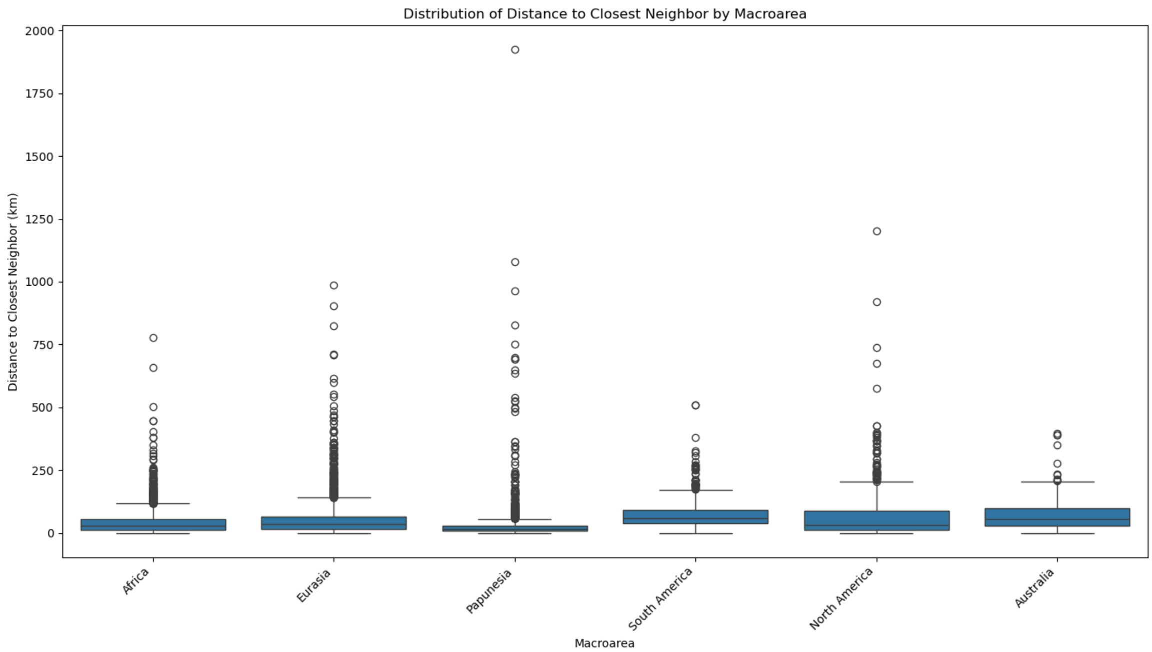
```
Checkpoint: Filtered out dialects. Remaining languages: 8604
Checkpoint: Starting to compute distances to closest neighbors.
/usr/lib/python3/dist-packages/pandas/core/arraylike.py:396: RuntimeWarni
ng: invalid value encountered in arccos
  result = getattr(ufunc, method)(*inputs, **kwargs)
Checkpoint: Finished computing distances to closest neighbors.
```




```
Top 10 Most Isolated Languages:
                            name     macroarea  distance_to_closest  \
412                        Aleut  North America          1202.010015
17870  Southern Cook Island Maori      Papunesia          1079.343213
5200                        Even        Eurasia           987.817891
3722          Cocos Islands Malay      Papunesia           962.130655
16506                      Sakha        Eurasia           905.173872
12290                    Moriori      Papunesia           827.764466
4206                     Dhivehi        Eurasia           824.751427
6060                     Guanche         Africa           777.280709
5845                   Gilbertese      Papunesia           750.731709
20875  Western Canadian Inuktitut  North America           738.943205

        latitude   longitude
412    52.122800 -174.290000
17870 -21.230000 -159.780000
5200   70.668700  130.914000
3722  -12.193342   96.833679
16506  61.697440  133.980310
12290 -44.000000 -176.500000
4206    1.928498   73.544330
6060   28.000000  -15.500000
5845    0.179000  173.640000
20875  64.348600  -96.148000
```

gready calculation with dialects in the distence calculation

In [19]:
```
def closest_neighbor_distance(language, languages):
    # print(f"Processing language: {language['name']}")
    min_distance = float('inf')
    for index, other_language in languages.iterrows():
        if language['name'] != other_language['name'] and \
           not pd.isna(language['latitude']) and not pd.isna(language['lo
           not pd.isna(other_language['latitude']) and not pd.isna(other_

            dist = haversine(language['latitude'], language['longitude'],
                             other_language['latitude'], other_language['
            min_distance = min(min_distance, dist)
    # print(f"Finished processing language: {language['name']}")
    return min_distance if min_distance != float('inf') else None

languages['distance_to_closest'] = languages.apply(closest_neighbor_dista

plt.figure(figsize=(14, 8))
sns.boxplot(x='macroarea', y='distance_to_closest', data=languages, showf
plt.title('Distribution of Distance to Closest Neighbor by Macroarea')
plt.xlabel('Macroarea')
plt.ylabel('Distance to Closest Neighbor (km)')
plt.xticks(rotation=45, ha='right')
plt.tight_layout()
plt.show()

isolated_languages = languages.sort_values(by='distance_to_closest', asce
print("\nTop 10 Most Isolated Languages:")
print(isolated_languages[['name', 'macroarea', 'distance_to_closest', 'la
```



```
Top 10 Most Isolated Languages:
                            name      macroarea  distance_to_closest  \
16093                    Rapanui      Papunesia          1923.370653   
412                        Aleut  North America          1202.010015   
17870  Southern Cook Island Maori      Papunesia          1079.343213   
5200                        Even        Eurasia           987.817891   
3722         Cocos Islands Malay      Papunesia           962.130655   
12835                    Naskapi  North America           920.835903   
16506                      Sakha        Eurasia           905.173872   
12290                    Moriori      Papunesia           827.764466   
4206                     Dhivehi        Eurasia           824.751427   
6060                     Guanche         Africa           777.280709   

        latitude   longitude  
16093 -27.113000 -109.342000  
412    52.122800 -174.290000  
17870 -21.230000 -159.780000  
5200   70.668700  130.914000  
3722  -12.193342   96.833679  
12835  55.931600  -61.131800  
16506  61.697440  133.980310  
12290 -44.000000 -176.500000  
4206    1.928498   73.544330  
6060   28.000000  -15.500000  
```