GeoTwins: Uncovering Hidden Geographic Disparities in Android Apps

Marco Alecci
SnT, University of Luxembourg
Luxembourg
marco.alecci@uni.lu

Pedro Jesús Ruiz Jiménez
SnT, University of Luxembourg
Luxembourg
pedro.ruiz@uni.lu

Jordan Samhi
SnT, University of Luxembourg
Luxembourg
jordan.samhi@uni.lu

Tegawendé F. Bissyandé
SnT, University of Luxembourg
Luxembourg
tegawende.bissyande@uni.lu

Jacques Klein
SnT, University of Luxembourg
Luxembourg
jacques.klein@uni.lu

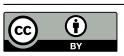
Abstract

While mobile app evolution over time has been extensively studied, geographical variation in app behavior remains largely unexplored. This paper presents a large-scale study of location-based Android app differentiation, uncovering two underexplored phenomena with significant security and fairness implications.

First, we introduce the concept of *GeoTwins*: apps that are functionally similar and share branding, yet are released under different package names (i.e., as two separate apps) across different countries. We investigate whether these seemingly identical apps diverge in critical aspects such as requested permissions, third-party libraries, code (i.e., smali files), and privacy disclosures. Second, we investigate the Android App Bundle ecosystem and examine whether supposedly consistent `base.apk` files remain invariant across regions, or if hidden regional customization exists that may affect app behavior or security.

To analyze these phenomena, we developed a distributed app collection pipeline spanning multiple regions and analyzed thousands of apps. We also release our dataset of 81 963 GeoTwins to facilitate further research. Our analysis reveals that around 20% of GeoTwin pairs exhibit meaningful differences, with the largest discrepancies occurring in the code, as well as in permissions, third-party libraries, and privacy disclosures. Furthermore, we find that nearly one in ten apps (8.6%) with the same package name exhibit significant cross-region differences in their `base.apk`, despite expectations of consistency.

These discrepancies have concrete consequences for researchers, developers, and regulators. As geographically distinct app variants can differ in their `base.apk`, the same app may be considered benign in one malware detection research study but potentially malicious in another, depending on where it was downloaded. At the same time, GeoTwins raise concerns about transparency and geographical fairness, as visually identical apps may differ in permissions, libraries, or privacy disclosures without users being aware.



MobiSys '26, Cambridge, United Kingdom

ACM ISBN 979-8-4007-2027-7/2026/06
https://doi.org/10.1145/3745756.3809191

ACM Reference Format:
Marco Alecci, Pedro Jesús Ruiz Jiménez, Jordan Samhi, Tegawendé F. Bissyandé, and Jacques Klein. 2026. GeoTwins: Uncovering Hidden Geographic Disparities in Android Apps. In *The 24th Annual International Conference on Mobile Systems, Applications and Services (MobiSys '26), June 21–25, 2026, Cambridge, United Kingdom.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3745756.3809191

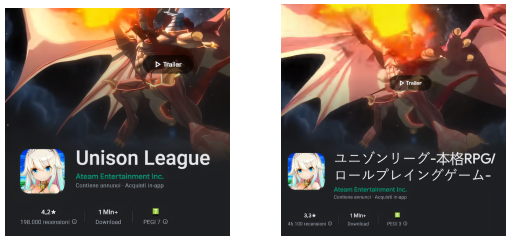
1 Introduction

Mobile applications (apps) shape digital experiences across the globe by providing access to communication, commerce, health, entertainment, and more. While extensive research has explored how apps evolve over time [12, 13, 32, 33, 37, 43], the orthogonal dimension of how app behavior varies by *location* has received far less attention. Recent studies have shown that Android apps may be unavailable in certain regions or offer region-specific content or permissions [27]. However, these studies primarily focus on app availability and often overlook the structural and functional divergence that can occur beneath the surface at the code implementation level. To address this gap, Guo et al. [22] conducted a comprehensive study of geo-feature differences in Android apps, revealing widespread variations in security-relevant aspects like advertising, data handling, and authentication across different countries. Their work with FREELENS, a novel framework designed to overcome challenges like code obfuscation, demonstrates that these regional code-level differences can frequently compromise security baselines and introduce disparities in privacy protections.

Our paper takes a step further. We address two unexplored aspects of regional differentiation in Android apps: ❶ We investigate the phenomenon of **GeoTwins** i.e., apps that are functionally similar, share a brand identity, but are released under different package names across different countries; and ❷ We analyze the **Android App Bundle** ecosystem and reveal unexpected differences in the `base.apk` files of apps, which are supposed to remain consistent across regions.

GeoTwins: Region-Specific App Variants. A central and novel finding of our study is the widespread practice of deploying functionally similar apps under *distinct package names* across countries. We refer to these regional variants as ***GeoTwins***. Despite appearing visually similar and sharing a brand identity, GeoTwins often differ in meaningful ways, such as their requested permissions,

third-party libraries, and privacy policies. For instance, multinational companies like *McDonald's* release country-specific versions of their app that differ in privacy disclosures and embedded services. Similarly, mobile games such as the game *Unison League* are published separately for Japanese- and English-speaking audiences under different package names: `jp.co.atm.unison` (Japan) and `en.co.atm.unison` (English) as seen in in Figure 1.



(a) en.co.atm.unison **(b) jp.co.atm.unison**

Figure 1: *Unison League* Google Play pages.

While this may seem normal in terms of language settings, resources, and images, our analysis revealed that the Japanese version of Unison League includes the `ACCESS_FINE _LOCATION` permission, whereas the international version does not. These cases highlight the subtle yet significant ways that app experiences and protections diverge based on geography. These discrepancies are nontrivial: they raise important concerns about transparency, user consent, and digital inequality. A user in one country may unknowingly receive a more privacy-invasive or less secure version of an app than a user elsewhere despite using what appears to be the same product as seen in Figure 1.

Moreover, even the average review score differs between the two versions, suggesting that regional variation may also affect user perception and highlighting the potential of GeoTwins as a framework for studying such differences in future work.

Regional Behavior via App Bundles. Additionally, the use of Android App Bundles [5] (a format mandated by Google Play since August 2021) enables developers to tailor app delivery based on users' regions and languages through dynamic delivery. By defining different configuration splits, developers can ensure that only the necessary components are downloaded based on the user's region or language. So while a `base.apk` file can remain common across all users, language-specific configuration APKs such as `config-en.apk` for English speakers or `config-ja.apk` for Japanese speakers can be delivered selectively. In theory, the `base.apk` should remain consistent across regions, with regional and language-specific differences handled through configuration splits. In our study, we tried to observe whether any differences existed in the `base.apk` files, as variations in split APKs are expected due to their configurable nature. Surprisingly, our analysis reveals that even the `base.apk` files can vary across countries when downloading the "same" app. These unexpected differences suggest deeper, region-specific customizations that may impact app logic, bundled services, or security practices, raising important questions about consistency and transparency in the app delivery pipeline.

Our Work. We have set up a crawling infrastructure across multiple regions and performed large-scale analysis on thousands of Android apps. In our analysis, we first examine the regional availability of the apps (similar to previous studies [27]), providing further insights on the matter. Then, we go beyond regional availability, structuring the analysis across three levels: ① First, we analyze general trends in location-specific apps (i.e., apps available exclusively in one region) to identify systematic differences between regions –for example, whether Japanese apps consistently differ from European ones–. ② Next, we examine apps with the same package name but downloaded from different locations, focusing on the `base.apk` component of Android App Bundles to detect region-specific variations introduced by the App Bundle mechanism. ③ Finally, we explore the GeoTwins phenomenon by identifying apps that maintain consistent branding and functionality but are distributed under different package names across countries, uncovering significant structural, behavioral, and privacy-related divergences.

Results & Implications. Our analysis reveals several key findings. Specifically, we observe that *nearly one in ten apps (8.6%) with the same package name exhibit non-trivial differences in their* `base.apk` across regions, despite expectations of consistency. Regarding GeoTwins, we found that *around 20% of pairs exhibit meaningful divergences*, with the largest differences occurring in the code, as well as additional variations in permissions, third-party libraries, and privacy disclosures. These results indicate that regional variation is not merely superficial but can affect core aspects of app behavior. These findings have concrete consequences for research and practice. They challenge the common assumption that a single app version represents the behavior of that app worldwide. For instance, consider an app X that was analyzed in a malware detection study and classified as benign. If a different version of X, downloaded from another country, includes additional permissions or third-party libraries, the same detection tool could potentially classify this regional variant as potentially malicious. Such location-specific divergences can silently bias malware detection, privacy analyses, permission modeling, and other empirical studies that implicitly rely on a single-country view of mobile apps. Beyond research, these results raise concerns about transparency and consistency, as visually identical apps may expose users to different privacy or security conditions depending on their location. We discuss these implications in detail in Section 5.

Contributions. This paper makes the following contributions:

- We conduct a **large-scale, cross-location analysis** that identifies systemic regional disparities in apps, with important implications for researchers, developers, users and regulators alike.
- We introduce the concept of **GeoTwins**: region-specific variants of the same app released under distinct package names, and demonstrate their functional and privacy-relevant differences.
- We investigate the Android App Bundle ecosystem and uncover **unexpected regional variations within `base.apk` files**, which are generally assumed to be uniform across locations.
- We release a curated dataset of 81 963 GeoTwins to facilitate further research in this area.

All our data and artifacts are publicly available here:

https://github.com/Trustworthy-Software/GeoTwins

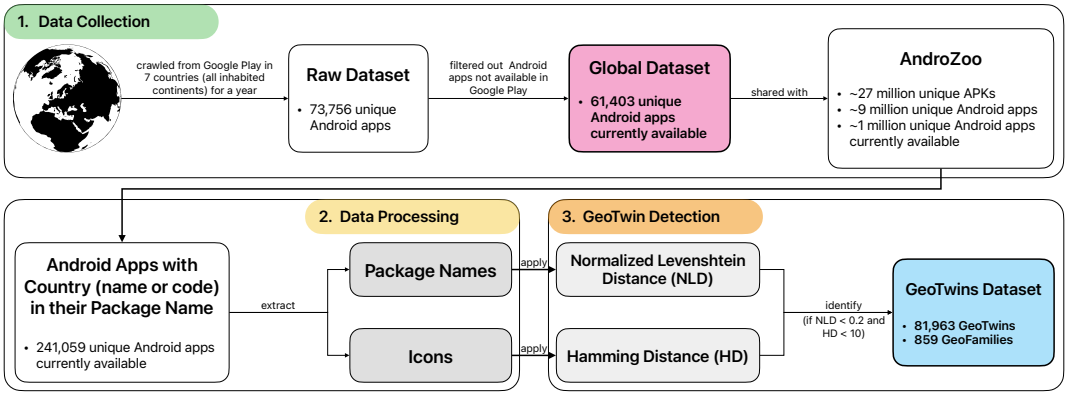


Figure 2: Overview of the data collection and processing pipeline used to construct the Global Dataset and the GeoTwins Dataset.

2 Background

This section outlines key concepts necessary to understand our study.

Core Terminology. We clarify the distinction between an *app*, an *APK file*, and a *package name*:

- *App*: An Android application software developed for Android OS, typically distributed via the Google Play Store, third-party markets, or sideloading.
- *APK file*: A zip-compressed binary archive that contains all components required to install and run an Android app, e.g., compiled code, resources, metadata, etc. Multiple APKs may exist for different versions of the same app.
- *Package name* (`pkgName`): A unique identifier (e.g., `com.example.myapp`) for each app, critical for installation, updates, and permissions. Two apps with different package names are treated as completely distinct by the Android operating system, even if they share identical branding or functionality.

App Bundle Format. In modern app distribution, developers must use the *App Bundle* format, which enables more efficient app delivery by tailoring APKs to device-specific needs (and therefore location-specific as well) [5]. This is explicitly requested by the official Android documentation with the following message: "Important: From August 2021, new apps are required to publish with the Android App Bundle on Google Play" [5]. An app bundle contains a base APK file along with zero or more split APKs.

- *Base APK*: The core component of an app that includes the essential code and resources required for the app to function. It should be common across all installations and is always included in the download. Its name is `base.apk`.
- *Split APK*: Optional APKs that handle device-specific configurations such as language, screen density, or architecture. These APKs are delivered selectively to match the user's device.

It is important to note that the structure of an Android App Bundle is determined at build time. In particular, whether certain resources (e.g., native libraries) are included in the base.apk is fixed for a given app version. This means that, under the standard App Bundle model, one device is not expected to receive a "large" base.apk while another receives a "thin" one; instead, devices may differ in the set of split APKs they are provided. In this work, we focus on the base.apk to isolate differences that cannot be attributed to standard configuration mechanisms.

3 Experimental Setup

In this section, we detail the experimental setup used throughout our study. For our analysis, we rely on two different APK datasets, each constructed using distinct methodologies. We now explain these datasets in detail. The first dataset (**Global Dataset**) comprises APKs sharing the same package name but obtained through location-specific crawling. The second dataset (**GeoTwins Dataset**) consists of GeoTwins APKs, i.e., regional variants of the same app with different package names across locations. Figure 2 provides an overview of the overall pipeline used to construct the datasets Finally, at the end of this section, we present our approach to pairwise app analysis, describing the features extracted via static analysis and the similarity scoring system we developed.

3.1 Global Dataset

In the literature, it is quite common to rely on the AndroZoo [3] dataset when working with Android apps, due to it being one of the largest collections of Android APKs available for researchers [1]. AndroZoo is reported to source its APKs from either Luxembourg, France, or Canada [3]. However, the exact location from which each individual APK was downloaded is not disclosed. Therefore, when downloading an APK from AndroZoo, it is not possible to determine whether it was retrieved from Luxembourg, France, or Canada.

As depicted in the first step ("Data Collection") of Figure 2, we decided to overcome this limitation and construct a more globally representative dataset by crawling APKs from the Google Play Store during one year, from seven different locations: Los Angeles, Santiago, Luxembourg, Johannesburg, Tel Aviv, Tokyo, and Sydney. These locations were carefully chosen to ensure broad coverage across all inhabited continents, with additional representation of large or diverse regions. For example, North and South America were represented by Los Angeles and Santiago, respectively, while Asia was covered by both Tokyo (Far East) and Tel

Aviv (Middle East). We deployed multiple virtual private servers (VPSs) via Vultr.com [41], a provider known for its extensive global VPS presence. Each of these VPSs hosted an independent crawler, which leveraged an unofficial Google Play API [9] to continuously search for and download APKs. To implement our crawlers, we adopted the same strategy used by the official AndroZoo crawlers. Additionally, the crawled APKs were submitted to AndroZoo to contribute to their repository and help consolidate resources for the research community.

Over the course of the year, this process yielded a total of 73 756 unique apps from across the globe (i.e., the "Raw Dataset" in Figure 2).

Although the APKs were downloaded from these varied locations, this does not necessarily imply that the apps are exclusively available in those countries, as many Google Play apps are globally accessible. To precisely determine the regional availability of each app, we implemented a verification procedure based on a reliable difference in the Google Play Store HTML page. Specifically, when you are logged in Google Play and an app is restricted in a given country, the "Install" button is omitted from its Google Play page. As shown in Figure 3, the presence of the "Install" button indicates the app is available for download in that region. By detecting this button in the HTML response, we can infer the app's availability status.

(a) App available **(b) App not available**

Figure 3: Google Play app page differences: (a) "Install" button present when the app is available in the region, (b) "Install" button absent when the app is not available.

To determine the availability of each app in our dataset, we followed this procedure:

(1) Establish a VPN connection corresponding to each of the seven original download locations.
(2) Send a request to the Google Play Store URL for each app: https://play.google.com/store/apps/details?id=PKG_NAME
(3) Inspect the HTML response for the "Install" button. If present, the app is marked as available in that region; if absent, it is marked as unavailable. A "404" response indicates that the app is no longer listed on Google Play.

This approach enabled us to generate a metadata file annotating each app package with its availability status across the seven surveyed locations. We then removed apps that were no longer available at the time of the availability check, i.e., those returning a 404 error. The resulting global dataset contains 61 403 unique apps (i.e., package names) associated with their APK files and comprehensive regional availability metadata.

3.2 GeoTwins Dataset

To collect GeoTwins, we employed a different approach that enabled us to gather a larger volume of apps within a shorter timeframe. While package names can sometimes hint at related apps, such as the *Unison League* game, with `jp.co.atm.unison` ("jp" for Japan) and `en.co.atm.unison` ("en" for English), this alone is insufficient for robust GeoTwins identification. To improve accuracy, we decided to incorporate an additional heuristic based on the visual similarity of app icons. Our intuition is that apps released in different countries but developed by the same developer typically share not only similar package names but also identical icons (as seen with *Unison League*) or visually similar icons, as illustrated in Figure 4.

(a) com.cisana.guidatv.de **(b) com.cisana.guidatv.at**

Figure 4: *Cisana TV+* app icons associated to different countries.

We started by relying on AndroZoo, now including the APKs from our "Global Dataset", as we did not need to know the download location to detect GeoTwins through our package name heuristic. Therefore, we began by extracting all distinct package names (8 887 673 in total) from the AndroZoo dataset. As expected, many of the apps from AndroZoo were no longer available on Google Play as of May 2025 and were therefore excluded. After removing these unavailable apps, we were left with 1 138 655 package names. To improve the reliability of the dataset and focus exclusively on region-specific variants, we applied an additional filter: we retained only those apps whose package names contained a country name or country code. This filtering step reduced the dataset to 241 059 package names, helping ensure that the resulting matches could be confidently associated with different geographical regions.

These 241 059 apps containing a country code/name represent the initial step of our "Data Processing" and "GeoTwin Detection" ss shown in Figure 2. Our goal was to identify potential GeoTwins within this set. To do this, we used the Google Play Scraper [21] to retrieve the current app icons for each package and computed their hash values using the difference hash (dHash) algorithm [26]. This algorithm is particularly effective for image comparison, as it captures visual similarities even when icons have undergone minor changes (e.g., resizing, color adjustments, or slight redesigns), such as in the example in Figure 4. Notably, we chose not to use icons extracted from the APK files available in AndroZoo since these files often correspond to outdated app versions and may not reflect the apps' current appearance on Google Play. For example, as shown in Figure 5, while older versions of *WhatsApp* found on AndroZoo had a distinct icon, recent updates show a revised design, which highlights the importance of using up-to-date visual data for accurate identification.

(a) *WhatsApp* icon from an APK found in AndroZoo. **(b) *WhatsApp* icon downloaded from Google Play.**

Figure 5: *WhatsApp* icon from an older version compared to the latest version.

In addition to the icons, we also compared package names to identify GeoTwins. Specifically, we used the normalized Levenshtein distance (NLD), a variant of the Levenshtein distance [28] obtained by dividing the Levenshtein distance by the longest string length, for comparing package names and the Hamming distance [23] for comparing icon hashes. We had to rely on NLD in addition to differences in country names or codes, due to cases like the GeoTwin `sk.martinus.knihovratok` and `cz.martinus.knihovratek`. These package names present slight differences from language variations ,"knihovrátok" in Slovak and "knihovrátek" in Czech, both meaning "bookworm". Given the vast number of possible pairwise comparisons (i.e., 29 054 600 211 pairs for the 241 059 packages available), we employed a Nearest Neighbors approach [16] to significantly reduce computation time. For icon similarity, we set a Hamming distance threshold of 10, consistent with prior studies [45, 47]. For string similarity, we adopted a normalized Levenshtein distance threshold of 0.2. While the optimal threshold can vary by context and would ideally be determined through extensive manual analysis (an effort beyond the scope of this study), we drew guidance from previous research [39, 44], which recommends values between 0.125 and 0.3. We selected 0.2 as a balanced choice: low enough to reduce false positives, yet permissive enough to capture the naming variations commonly associated with region-specific versions. Finally, we retained only those pairs with distinct country names or codes to further ensure that the apps targeted different regions. Although this filtering step significantly reduced the number of potential matches, it was a necessary trade-off to produce a high-confidence dataset. The resulting GeoTwins dataset contains 81 963 GeoTwins, i.e., is 81 963 pairs of regional variants of the same app, released under different package names.

GeoFamilies. Once the GeoTwins dataset was created, we observed that many GeoTwins could actually be grouped into larger collections of related apps. These collections, which we refer to as *GeoFamilies*, consist of multiple regional variants of the same underlying app. Each variant is typically targeted to a different country and published under a slightly different package name, often with minor localization adjustments. For example, as shown in Table 1, the GeoFamily for *Unison League* includes `jp.co.atm.unison` and `en.co.atm.unison`, while the *Papa John's* app includes variants such as `cl.com.papajohns`, `pa.com.papajohns`, and `cr.com.papajohns`. A larger example is the *American Express* app family, which spans nine different regional variants.

Table 1: Examples of GeoFamilies.

GeoFamily	Package Names
Unison League	en.co.atm.unison
	jp.co.atm.unison
Papa John's	cr.com.papajohns
	cl.com.papajohns
	pa.com.papajohns
American Express	com.americanexpress.android.acctsvcs.us
	com.americanexpress.android.acctsvcs.be
	com.americanexpress.android.acctsvcs.au
	com.americanexpress.android.acctsvcs.uk
	com.americanexpress.android.acctsvcs.nl
	com.americanexpress.android.acctsvcs.de
	com.americanexpress.android.acctsvcs.ca
	com.americanexpress.android.acctsvcs.fr
	com.americanexpress.android.acctsvcs.japan

The GeoTwins dataset we built contains a total of 81 963 GeoTwins. These pairs can be grouped into 859 distinct GeoFamilies with a median number of 2 distinct package names per GeoFamily and an average of 8.82 package names. It is noteworthy to highlight that within a GeoFamily of n apps, all unique pairs of apps (i.e., combinations of two apps) are considered GeoTwins using the following formula:

$$\#\text{GeoTwins} = \binom{n}{2} = \frac{n \times (n-1)}{2}$$

For example, a family with three apps a, b, and c yields three pairs of GeoTwins: (a, b), (a, c), and (b, c), as in the case of `com.papajohns`. While GeoFamilies offer a broader view of app release strategies across regions, this paper focuses exclusively on GeoTwins, i.e., individual pairs of region-specific variants. To avoid redundancy and ensure statistical independence, we consider at most one GeoTwins pair per GeoFamily in our analysis (e.g., in RQ4). A comprehensive study of full GeoFamilies ,including their internal structure, evolution over time, and localization strategies, is left for future work.

3.3 Apps Analysis

Our analysis focuses on pairwise comparisons between APKs. Specifically, we compare APKs with the same package name but downloaded from different locations (Global Dataset), as well as pairs of GeoTwins APKs (GeoTwins Dataset). Due to the large volume of apps and the associated time and resource constraints, we decided to rely on static analysis techniques. To perform these comparisons, we developed a custom similarity framework in Python, built on top of ApkTool[8] and Androguard[4]. Our tool extracts a comprehensive set of features from each APK, including: the list of permissions requested; the names of app components (activities, services, receivers, providers); the APK certificate; third-party libraries used; native libraries; hard-coded URLs embedded within the APK; and all files contained in the APK, with particular emphasis on the ".smali" files. Smali files are the human-readable representation of Dalvik bytecode, the intermediate format executed by the Android runtime. We selected these features because they collectively capture both high-level app behavior (e.g., permissions, components, libraries) and low-level code differences (e.g., smali code, embedded files). This broad set of features enables us to detect both functional and structural variations that may arise from regional customizations or divergent development practices.

Although existing tools such as SimiDroid [30] have been developed to compute similarity between Android APKs, they primarily operate at a relatively coarse granularity, focusing on method-level code comparison, component-level analysis, and resource-based similarity [30]. In contrast, our framework is designed for fine-grained, feature-level comparison tailored to identifying regional differences. In particular, our analysis requires capturing subtle variations in permissions, third-party libraries, embedded URLs, certificates, and file-level differences (including smali code), which may not significantly affect overall structural similarity but are critical for understanding region-specific behavior. This design allows us to uncover region-specific divergences that would otherwise remain hidden when relying on SimiDroid. Similar to SimiDroid, we compute a similarity score for each extracted feature, as well as an overall similarity score that aggregates feature-level comparisons. Each score ranges from 0 (completely dissimilar) to 1 (identical).

Our similarity tool is publicly available in our repository and can be used to compare any pair of APKs, representing an additional contribution to this work.

Similarity Score. We compute feature-level similarity scores using metrics suited to each feature type:

- **Set-based features (permissions, components, libraries, URLs):** Similarity is measured using the Jaccard index [24]:

$$J(A_1,A_2)=\frac{|A_1\cap A_2|}{|A_1\cup A_2|}.$$

- **Files and Smali files:** For all files (and separately for `.smali` files), we compute a modified Jaccard score. Files with the same name and hash are treated as identical, and those with the same name but different hashes as similar (weight 0.5):

$$J_{\text{mod}}=\frac{|F_{\text{identical}}|+0.5|F_{\text{similar}}|}{|F_1\cup F_2|}.$$

- **Certificates:** After flattening the JSON structure, certificate similarity is the fraction of matching key,value pairs:

$$S=\frac{\text{matching key-value pairs}}{\text{unique keys}}.$$

The overall similarity score is the average of all feature scores, and our tool reports both aggregated and feature-level values.

4 Evaluation

Our study is guided by four research questions that examine both the availability and internal composition of Android apps across different geographical regions.

- **[RQ1] Regional Availability:** How does the availability of Android apps vary across different locations?
- **[RQ2] General Trends Across Regions:** Are there systematic differences in Android apps across geographical regions (e.g., Japanese apps vs. European apps)?
- **[RQ3] Location-Dependent Differences in Base APKs:** For apps sharing the same `packageName`, do the `base.apk` files delivered by the Play Store differ based on the user's geographical location? If so, what are the nature and implications of these differences?
- **[RQ4] GeoTwins Divergence:** Do GeoTwins exhibit differences in structure, security practices, or other behaviors? If so, what are these key differences?

4.1 [RQ1] Regional Availability

In this research question, we examine the regional availability of apps within the Global Dataset introduced in Section 3. Using the metadata collected during our availability checks, we assess how app accessibility varies across the seven selected geographical locations. Table 2 presents a quantitative summary of app availability. Our dataset comprises 61 403 unique package names, each verified for presence in the Google Play Store across all seven regions.

As shown, the majority of apps, specifically 48 178 (78.46%), are available in all surveyed locations, suggesting strong global distribution across much of the mobile app market. In contrast, 10 685 apps (17.40%) are exclusive to only one of the seven locations. This highlights a significant subset of regionally restricted apps, indicating the influence of location- or culture-specific targeting, regulatory constraints, or localization strategies. It is important to note that when we refer to apps available in only "one location", we mean strictly within the seven locations included in our analysis. For example, an app available "only" in Tokyo (in the context of our study) may also be accessible in other countries not included in our dataset (e.g., South Korea). Nonetheless, this methodology provides meaningful comparative insights into the geographical availability of Android apps across a diverse and representative set of global locations.

Table 2: App availability across the seven inspected locations.

Number of Locations	Number of Apps	Percentage
7	48 178	78.46%
6	779	1.27%
5	179	0.29%
4	198	0.32%
3	370	0.60%
2	1007	1.64%
1	10 685	17.40%

To better understand the characteristics of the 10 685 apps available in only a single location, we analyzed their distribution across the seven locations analyzed. Figure 6 shows the number of apps exclusive to each location. Tokyo stands out, accounting for 5807 exclusive apps, far more than the next highest, Los Angeles, with 1718. This suggests that Tokyo hosts a particularly distinct app market. Contributing factors may likely include Japan's unique cultural context, which shapes user expectations around app features, UI/UX design, and content. Developers may thus create apps specifically for the Japanese market or publish heavily localized versions of global apps. Additionally, language complexity and high user standards may incentivize maintaining separate versions.

In summary, while most apps are broadly accessible, a significant minority are regionally exclusive, particularly in Tokyo, reflecting market dynamics driven by cultural and linguistic factors. Unlike prior work, such as Kumar et al., [27], which explored the reasons for regional restrictions, our focus is on quantifying and characterizing availability patterns across diverse regions. Investigating the causes of unavailability is beyond our current scope, as they have already addressed them in their study.

> **Answer to RQ1:** *App availability is largely global, with 78.46% of apps accessible in all seven locations studied. However, 17.40% of apps appear only in a single location ,most notably in Tokyo, indicating a significant degree of regional exclusivity, which could be influenced by differing cultural environments and user preferences.*

4.2 [RQ2] General Trends Across Regions

In this research question, we investigate whether systematic differences exist among Android apps across geographical regions by analyzing how the 10 685 apps exclusive to a single region (identified in RQ1 and shown in Figure 6) differ. Each region's apps are analyzed independently, without cross-location matching, allowing us to uncover high-level regional trends in design and behavior. This broad analysis serves as a foundation for the more detailed comparisons explored later in RQ3 and RQ4. While many technical and behavioral features could be examined, we focus on three key aspects: ❶ Privacy Policies, ❷ Third-Party Library Usage, and ❸

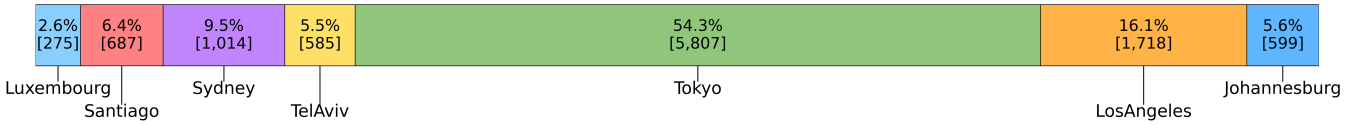


Figure 6: Distribution of apps available exclusively in one location.

Dangerous Permission requests. These were chosen for their clear relevance to privacy and app behavior, and to serve as illustrative examples of region-specific disparities.

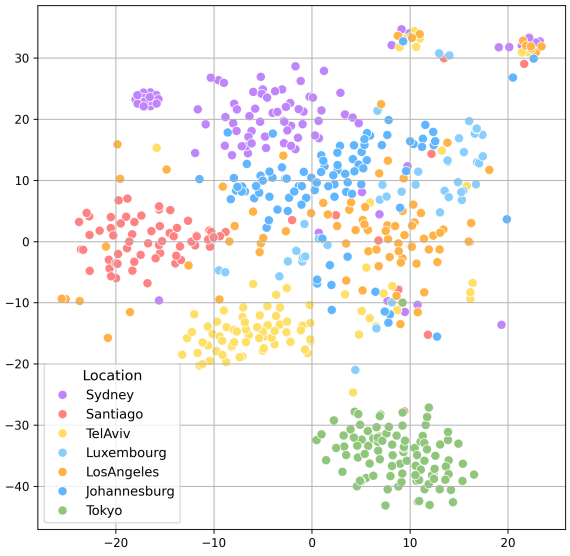


Figure 7: 2D visualization of privacy policy text embeddings across the seven selected locations.

❶ Privacy Policies. We begin by examining the content of privacy policies, as apps published on platforms like Google Play are typically expected to provide such policies to inform users about their data practices. We collected privacy policy URLs from the Google Play pages of location-exclusive apps. Due to the imbalance in app counts across regions (e.g., nearly 6000 in Tokyo vs. 275 in Luxembourg), we used a controlled sampling strategy, selecting the top 100 most downloaded apps per region within our dataset, provided they had a valid privacy policy URL. This ensures focus on popular apps representative of typical user experience. We then translated all non-English policies to English using the DeepL Pro API [17], embedded them using OpenAI's `text-embedding-3-small` model [35], and visualized them via t-SNE [40] (a dimensionality-reduction technique). Figure 7 shows the resulting 2D scatterplot.

The plot reveals location-specific clustering of privacy policy content; for instance, policies from Tokyo or Tel Aviv form distinct groups, suggesting these documents reflect regional norms, expectations, and regulatory environments rather than adhering to a uniform global standard. We further discuss potential explanations for these differences in Section 5.

To complement this aggregate view, we manually inspected a small subset of privacy policies across regions. For instance, we observed that some apps distributed in Luxembourg tend to provide more detailed and explicit descriptions of data usage and user rights, whereas policies from other regions are often more generic. A comprehensive manual analysis of privacy policies, however, would require substantial effort due to their natural-language complexity and legal nuances and is therefore beyond the scope of this paper, which focuses on scalable, automated analysis.

❷ Third-Party Library Usage. Third-party libraries are widely used to implement functionality such as ads, analytics, networking, and UI. However, they also pose risks related to tracking, data sharing, and compliance. Using again a sample of 100 apps per region, we extracted the list of third-party libraries used. We then computed normalized usage frequencies for each region and compared them with global averages. Figure 8 presents a heatmap showing location-based deviations for the 30 most common libraries.

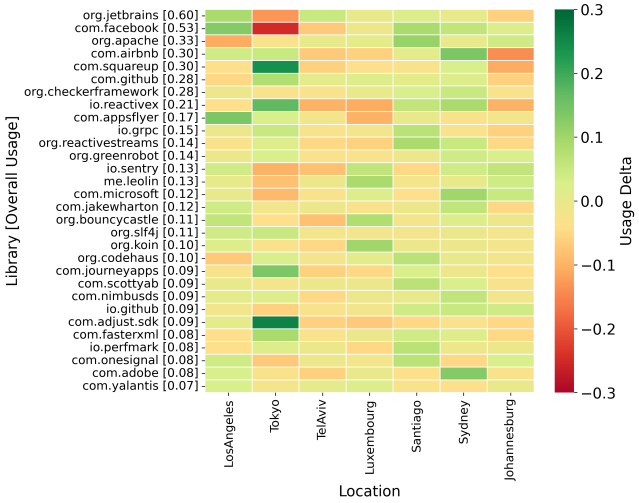


Figure 8: Location-based deviations in the usage frequency of the top 30 third-party libraries.

Several patterns emerge. For example, `com.facebook` is more prevalent in Los Angeles and less so in Tokyo, likely reflecting differences in integration with Western versus Eastern platforms, which will be discussed extensively in Section 5.

❸ Dangerous Permissions. We also analyzed the use of Android permissions, focusing on both total permissions and those deemed "dangerous" by the Android documentation[6],i.e., those granting access to sensitive data or control over critical device features. Using the 100-app-per-location sample again, we extracted requested permissions as described in Section 3.3. We calculated the average number of permissions (dangerous and total) for each location and

compared these values to the global average (across all locations). The results are shown in Figure 9.

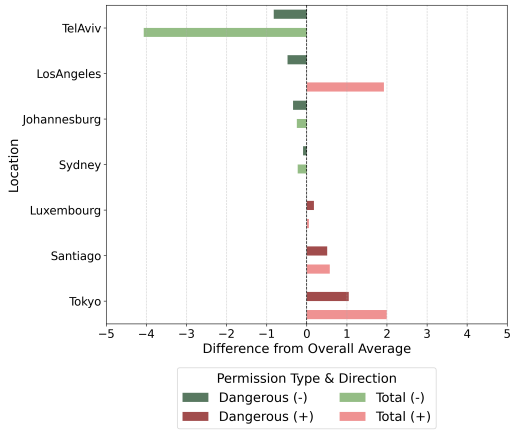


Figure 9: Average deviation from global average in requested (dangerous) permissions, by location.

As shown, Tokyo apps request more total and dangerous permissions than the global average, while Tel Aviv apps request fewer. Other locations show milder deviations. These differences suggest that regional development styles or regulatory factors also influence how apps engage with the Android permission model. We further discuss potential explanations for these differences in Section 5.

To complement this aggregate view, we also examined the most frequently requested dangerous permissions across regions. We observed recurring patterns in the types of permissions requested. For instance, the `CAMERA` permission consistently appears among the top five most requested dangerous permissions across all locations except Tel Aviv. These observations suggest that regional differences are not only reflected in aggregate statistics, but also in the relative prominence of specific sensitive permissions. At the same time, such patterns may also reflect the distribution of popular app types in each region. A more fine-grained, app-level analysis would be required to disentangle these factors, which is beyond the scope of this work.

Answer to RQ2: *Apps across regions differ not only in availability (as seen in RQ1) but also in key structural and privacy-related aspects. Privacy policies cluster by location, library usage varies, and permission requests highlight the potential systemic influence of geography on Android app behavior.*

4.3 [RQ3] Location-Dependent Differences in Base APKs

This research question investigates whether apps with the same package name but downloaded from different regions,i.e., the apps from our **Global Dataset**, exhibit differences in their `base.apk` files. As it would be extremely time- and resource-consuming to analyze all the apps in our Global Dataset available in all seven locations (48 178 apps as presented in RQ1), we decided to analyze a statistically significant random sample of apps. Using a 95% confidence level with a 5% margin of error, we computed a sample size of 382 apps (i.e., 764 APKs). While we initially collected apps during the dataset construction, we had to ensure that the versions compared were the latest ones and that downloads occurred within a narrow time window, reducing the likelihood of version drift. We, therefore, re-crawled all sampled apps across all seven locations simultaneously. This approach is similar to the methodology employed by Kumar et al. [27] for collecting the applications used in their study. For each app, we downloaded the full app bundle, including the `base.apk` and split APKs, but then limited our analysis to the `base.apk` only. This allowed us to focus our comparison specifically on the `base.apk` component, avoiding the expected and acceptable differences found in configuration-specific split APKs (as described in Section 2). In doing so, our study addresses a key limitation in prior work [22, 27], which did not distinguish between base and split APKs. This oversight in earlier studies may have led to false positives by attributing configuration-driven differences ,intentionally introduced by Google Play, as evidence of behavioral variation. Using the custom similarity score tool introduced in Section 3.3, we conducted all 21 region pairwise comparisons for each app. For example, an app from Luxembourg was compared against its versions from the six other locations.

Results. Among the 382 sampled apps, 33 (~8.6%) showed differences in their `base.apk` files. While small in proportion, this is notable given that `base.apk` files are expected to be consistent across locations. We assessed these differences by computing similarity scores across their features (Table 3) and visualizing the score distributions (Figure 10).

Table 3: Similarity metrics between GeoTwins. N.A. indicates uniformity across all samples.

Feature	Average	Median	Correlation with Overall
Permissions	0.96	1.00	0.91
Components	0.97	1.00	0.92
Certificates	1.00	1.00	N.A.
Third-party Libs	0.96	1.00	0.91
Native Libraries	1.00	1.00	N.A.
URLs	0.93	0.99	0.89
Files	0.52	0.50	0.58
Smali Files	0.43	0.50	0.57
Overall	**0.84**	**0.87**	**1.00**

As shown, permissions, components, and third-party libraries are nearly identical across regions (avg/median 0.96–1.00), suggesting consistency. URLs are slightly more variable (avg 0.93), which may vary across regions. Files and Smali code exhibit the greatest variation (avg ~0.43–0.52), which largely accounts for overall discrepancies. Notably, 91% of the overall similarity scores fall between 0.80 and 0.99, with an average of 0.84 and a median of 0.87. We now provide a few illustrative examples while we later discuss potential explanations for these differences in Section 5.

Illustrative examples. Despite high similarity scores overall, some apps reveal substantial differences. For example, `club.eatery.prostopizza` shows a permission similarity of 0.74 between the Santiago and Luxembourg versions. The Luxembourg variant requests 26 permissions, including `USE_BIOMETRIC`, `USE_FINGERPRINT`, `READ_GSERVICES`, and `ACCESS_ADSERVICE_TOPICS`, while the Santiago version requests 25, with unique permissions such as `ACCESS_`

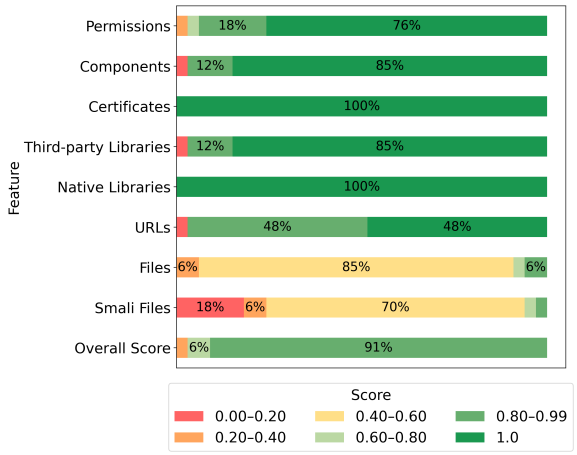


Figure 10: Distribution of similarity scores across features. Scores below 3% are omitted for readability.

BACKGROUND_LOCATION and READ_MEDIA_IMAGES. This highlights discrepancies in biometrics, location, and media access.

Another case, com.shipbuild.buildwishipgb.ckxm, shows differences in third-party libraries between its Los Angeles and Tel Aviv versions. The LA version includes 12 libraries, while the Tel Aviv one includes 11. Both share common libraries such as com.bytedance, com.squareup, and com.facebook, but the LA variant uniquely includes com.fyber, suggesting region-specific advertising configurations.

Answer to RQ3: *While most apps with the same package name have highly similar* base.apk *files across regions, around 8.6% exhibit non-trivial differences, particularly in files, code, and occasionally permissions or libraries. These findings challenge the assumption of a globally consistent* base.apk *and suggest hidden region-specific customizations in some apps.*

4.4 [RQ4] GeoTwins Divergence

The goal of this research question is to analyze the differences between GeoTwins and assess the extent of their divergence by examining the apps included in the GeoTwins Dataset, introduced earlier in Section 3. From our full GeoTwins dataset of 81 963 pairs, we selected a statistically significant sample using the same methodology as in RQ3. Based on a 95% confidence level and a 5% margin of error, this resulted in a sample of 383 GeoTwins (i.e., 766 APKs). To avoid sampling bias from large, overrepresented GeoFamilies (see Section 3.2), we sampled at most one GeoTwin pair per GeoFamily. The selected pair is chosen randomly to avoid introducing selection bias. This approach ensured both statistical validity and broad coverage across diverse regional variants.

Since the GeoTwins Dataset was derived from AndroZoo metadata, we could have used AndroZoo to retrieve the APK files. Nevertheless, we chose to download the APKs ourselves to ensure we obtained their latest versions and to avoid inconsistencies caused by potential updates since the time AndroZoo crawled them (AndroZoo does not publish the exact crawl date). This also allowed us to extract and analyze both the base.apk and split APKs, though our analysis here, like in RQ3, focuses on the base.apk. By collecting and analyzing GeoTwins, this work addresses a key limitation in Guo et al. [22], who compared apps with shared package names across multiple locations (similar to our approach in RQ3) but lacked the ability to compare what they referred to as "regional variants".

Results. We then compared the sampled GeoTwins using the custom similarity score tool introduced in Section 3.3 (similarly to what we did in RQ3). Table 4 reports the average, median, and correlation with the overall score for each feature. Figure 11 further illustrates the distribution of the similarity score for each feature.

Table 4: Similarity metrics between GeoTwins.

Feature	Average	Median	Correlation with Overall
Permissions	0.95	1.00	0.72
Components	0.94	1.00	0.80
Certificates	0.92	1.00	0.29
Third-party Libraries	0.93	1.00	0.78
Native Libraries	0.98	1.00	0.43
URLs	0.87	0.97	0.75
Files	0.87	0.97	0.78
Smali Files	0.76	0.98	0.83
Overall Score	**0.90**	**0.98**	**1.00**

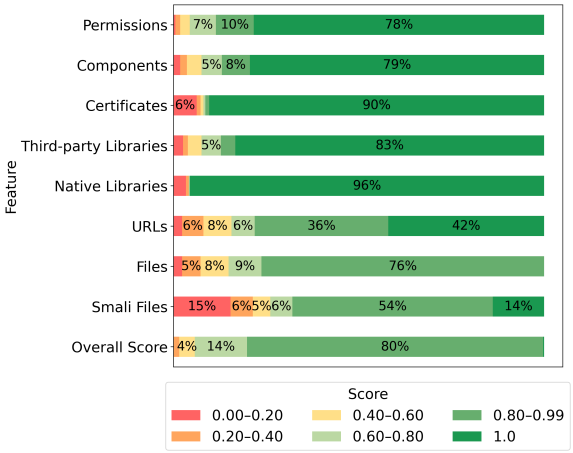


Figure 11: Distribution of similarity scores for different app features, categorized by score ranges. Values below 3% are omitted for better visualization.

As it can be observed, GeoTwins are largely similar across most features: permissions, components, libraries, and certificates have high median scores (1.0), indicating full similarity in at least half of all samples. The largest divergences appear in smali code (avg. 0.76) and files (avg. 0.87), which also have the strongest correlation with the overall score, suggesting code-level changes are the primary source of divergence.

In over 80% of GeoTwins, permissions, components, certificates, and third-party libraries are identical. Even in features with more variability, like smali files, files, and URLs, differences are relatively modest. Only 33% of smali file comparisons fall below a similarity

score of 0.8, and fewer than 25% of file or URL comparisons do. No GeoTwins are completely identical overall (due in part to changes in package name and signing), but nearly 35% have an overall similarity score above 0.99. This supports the hypothesis that GeoTwins are often minor variants of the same app, republished for regional targeting with limited underlying changes.

Our analysis indicates that most changes made to the APKs before republishing them in different regions are minimal. For instance, as seen in Figure 4, some applications are republished with different icons depending on the target country. These changes, while mostly cosmetic, affect the files similarity score, as the internal icon file is modified. More critically, we observed cases in which republished apps differ in their declared permissions and even in their components. This is particularly concerning because it raises questions about whether users in different countries are subject to different risks, despite apparently using the same application.

Illustrative Examples. To further explore this topic, we closely examined three examples from our dataset, which exhibited differences across some of their features. Building on this analysis, we later discuss potential explanations for these differences in Section 5.

In the case of the *Wine* app, with a permissions score of 0.952, we found that the version with the package name `br.com.wine.app` declares two additional permissions not present in `br.com.wine.app.mx`: `ACCESS_ADSERVICES_TOPICS` and `ACCESS_ ADSERVICES _CUSTOM_AUDIENCE`. These extra permissions are related to advertising, and their inclusion in a single APK raises questions about their underlying motivation.

Meanwhile, in the case of the *Knihovrátok* app (or *Knihovrátek*, depending on the package name), which has a components score of 0.889, we discovered that the version with the package name `sk.martinus.knihovratok` declares some additional components not found in `cz.martinus.knihovratek`. Specifically, it includes one extra service and two extra content providers: the service is `com.baseflow. geolocator.Geolocator.LocationService`, and the providers are `com.crazecoder.openfile.FileProvider` and `androidx. core.content.FileProvider`. The inclusion of the geolocation service is particularly concerning, as it may allow the Slovak version of the app to access users' location data, a feature apparently absent in the Czech version. Again, these discrepancies raise concern about the motivations behind publishing regional variants.

Finally, we have the example shown in Figure 1. The *Unison League* GeoTwins have an overall similarity score of 0.891, with notable differences across most features, especially in certificates and permissions. It is uncommon to find different certificates among GeoTwins, but closer inspection reveals that the Japanese version is developed by a different team within the same company (*Ateam*[1]): the *ito* team for Japan and the *imai* team for the international version. Additionally, as noted in Section 1, the Japanese version requests the `ACCESS_FINE_LOCATION` permission, while the international version includes `MOUNT_UNMOUNT_FILESYSTEMS`. These differences suggest permissions are adapted to comply with region-specific regulations (e.g., GDPR [19]) rather than solely the game's needs.

[1]https://www.ateam-entertainment.com

Answer to RQ4: *GeoTwins are largely similar across most features. Our analysis shows that over 80% of GeoTwin pairs share identical permissions, components, certificates, and third-party libraries. Differences mainly appear in smali code, URLs, and other files, though these are generally minor. Nearly 35% of GeoTwins have a similarity score above 0.99, supporting the idea that these apps are mostly repackaged with small tweaks for regional use. Permission differences likely reflect local regulatory compliance rather than functional changes.*

5 Discussion

Our findings reveal that Android apps exhibit substantial, often hidden, regional variation at both the binary and behavioral levels. In this section, we discuss ① the **potential motivations** behind these geographical differences and ② their **broader implications** for research, development, and regulation.

5.1 Motivations behind geographical differences

While our analysis does not establish causal relationships, our observations suggest several plausible factors that may explain the regional divergence observed in Section 4.

Regulatory and Compliance Requirements. Regional regulations, such as privacy or advertising laws, may influence app behavior, including permission usage and the integration of specific services. For instance, apps distributed in regions with stricter data protection regulations may limit certain tracking functionalities or request fewer permissions. A representative example emerges from our analysis of two regional variants of the Wine application (`br.com.wine.app` and `br.com.wine.app.mx`). The Brazilian variant declares additional permissions related to Android's Privacy Sandbox advertising APIs (i.e., `ACCESS_ADSERVICES_TOPICS` and `ACCESS_ADSERVICES_CUSTOM_AUDIENCE`), which enable interest-based profiling and remarketing. Consistently, its privacy policy (subject to the Brazilian *Lei Geral de Proteção de Dados* (LGPD) [10]) provides a more explicit description of personalized advertising, user profiling, and data usage for targeted marketing. In contrast, the Mexican variant does not include these permissions, and its corresponding privacy policy (governed by the Mexican *Ley Federal de Protección de Datos Personales en Posesión de los Particulares* (LFPDPPP) [18]) is less explicit regarding profiling and advanced advertising practices, focusing instead on general marketing purposes and user rights (e.g., ARCO rights). While this example does not establish a direct causal relationship, it illustrates how regional differences in technical behavior can align with variations in regulatory context and privacy disclosures.

Regional Ecosystem Preferences. Differences in local ecosystems can influence the adoption of third-party libraries and services. For example, certain analytics, advertising, or social media SDKs may be more prevalent in specific regions, leading to systematic differences in embedded libraries across app variants, as observed in Section 4.2. In the case of Japan, prior studies and reports indicate that platforms such as LINE and YouTube dominate social media usage, while Facebook is comparatively less popular [34, 38]. This suggests a reduced incentive for developers targeting the Japanese market to integrate Facebook-related SDKs, which is consistent

with our findings in Section 4.2, where the `com.facebook` library is less prevalent in apps from Tokyo.

Technical Differences. Regional variants of the same app may differ in their underlying technical configurations, including backend services or communication protocols. These differences can lead to inconsistencies in security-relevant aspects, even when apps share branding and functionality. For example, we observed two regional variants of a city exploration app, `org.mapapps.mapyourtown.turkey` and `org.mapapps.mapyourtown.skorea`, which rely on different backend endpoints. The Turkish version communicates with a server over HTTP (http://keos.atakum.bel.tr), while the South Korean version uses HTTPS (https://www.gijangcmc.or.kr). While this difference may stem from technical constraints, it illustrates how regional variants can introduce inconsistencies in secure communication practices. Such variations may also reflect decentralized development processes, where different regional teams maintain separate backend infrastructures or deployment pipelines, as observed in apps such as Unison League (see Section 4.4), which are released as distinct regional variants.

Overall, while these factors provide plausible explanations, they remain indicative rather than definitive. These differences are likely the result of a combination of regulatory, technical, and social influences, rather than a single underlying cause. Hence, fully understanding the motivations behind them would require interdisciplinary approaches. In this regard, prior work in sociotechnical studies has emphasized that mobile apps should be understood as components of broader digital ecosystems, shaped by contextual factors rather than isolated design choices [11]. For instance, a longitudinal analysis of the Kazakhstani Kaspi app shows how its integration of third-party services evolved over time, with a growing reliance on SDKs associated with Chinese platforms, reflecting shifts in its underlying ecosystem and external dependencies [11].

5.2 Implications of geographical differences

While our analysis focuses on identifying regional differences, these findings also have broader implications for multiple actors.

❶ **Implication for researchers.** A recurring assumption in prior work on Android apps is that a single version of an app is representative of all its versions. Our results challenge this assumption: apps frequently exhibit region-specific behaviors, permissions, and libraries. As a result, research that relies on a single country-specific version, such as studies of permission evolution or privacy leakage, may suffer from geographic bias. For example, a benign version of an app in one region might request significantly fewer permissions than its counterpart in another, potentially leading to misclassification or skewed conclusions. Indeed, even small variations in features (e.g., permissions or API calls) can impact malware detection outcomes, as shown by prior work [14, 15].

Similarly, the reproducibility of existing studies can also be undermined. Silent, location-based variations in APKs mean that reproducing prior results may inadvertently analyze a different regional variant, making conclusions non-replicable. Our dataset enables the research community to explicitly account for such regional differences.

Additionally, many longitudinal analyses [13, 20, 29, 46] track apps by package name, implicitly assuming that a package corresponds to a single product. The concept of *GeoTwins* shows that this assumption can be misleading: the same brand may be represented by multiple package names across different countries, and even apps sharing the same package name may exhibit regional differences in their `base.apk`. These findings suggest that future analyses should consider location, enabling a more comprehensive understanding of cross-country variation in privacy, advertising, and functionality.

❷ **Implications for Users and Regulators.** GeoTwins raise important questions about transparency and informed consent. Users downloading visually identical apps from different regions may unknowingly receive versions with more invasive permissions, less robust privacy safeguards, or different embedded services. For example, we identified regional variants of the same app (e.g., `com.appmind.radios.za` and `com.appmind.radios.be`) that share the exact same privacy policy on Google Play, yet differ in their requested permissions: the South African version requests `android.permission.BLUETOOTH_ADMIN`, while the Belgian version does not. This discrepancy is not reflected in the privacy policy, suggesting that users may be exposed to additional capabilities without being explicitly informed. Regulators could consider requiring platforms to disclose regional customizations more explicitly to ensure consistent user protections and to enforce compliance with local regulations.

❸ **Implications for Developers.** Our findings are relevant for developers, who should take into account regional differences driven by regulatory requirements, as illustrated by the Wine app example. At the same time, our dataset can be used to gain insights into the libraries and services that are most commonly adopted in different regions, enabling developers to better align their apps with local ecosystem preferences, as observed in our analysis of third-party libraries (e.g., the varying prevalence of the `com.facebook` SDK across regions). For instance, a promising direction for future work is to leverage our dataset as a starting point to collect and analyze user reviews across regions, enabling the study of how users in different countries perceive and react to seemingly identical apps. Such insights could help developers better understand how regional variations in implementation influence user experience, satisfaction, and trust.

Overall, our work highlights the need to rethink how research, regulation, and development practices account for regional app differences, and provides the first large-scale dataset to support such efforts.

6 Limitations

In this section, we acknowledge the potential limitations of our study, explain how we mitigated them, and suggest how these challenges can be addressed in future research.

Motivations and implications. While our discussion provides plausible explanations and illustrative examples of regional differences, we emphasize that a comprehensive understanding of their underlying causes would require a deeper, case-by-case analysis of individual apps. In particular, explaining why a specific permission or behavior differs across regions would involve inspecting backend services, business logic, third-party integrations, and regulatory

constraints. Such an investigation would require extensive manual effort and is therefore beyond the scope of this work. As a result, our analysis focuses on identifying and characterizing the presence of regional differences at scale, rather than establishing causal relationships. We view this as a first step that motivates more targeted, in-depth investigations. As part of future work, we plan to further explore the practical impact of these differences, for example, by evaluating how region-specific app variants affect the behavior and consistency of Android malware detection tools.

Geographical Coverage. The list of countries included in our study is not exhaustive and may not fully represent the global diversity of app ecosystems. To mitigate this limitation, we selected countries from different macro-regions (e.g., Asia, North America, Europe, and Africa) to provide a broad, albeit non-comprehensive, geographic perspective. Our data collection relies on geographically distributed VPS infrastructure, and it is not always feasible to obtain reliable and stable instances in all countries. In particular, deploying infrastructure in emerging markets can be more challenging due to limited availability and higher operational constraints. As a result, while we aim for broad coverage, certain regions remain underrepresented.

Static Analysis Only. Our methodology relies solely on static analysis due to the large volume of apps examined. While static techniques offer scalability and reproducibility, they cannot capture runtime behaviors such as dynamic permission requests or API calls. Future work may incorporate dynamic analysis tools, such as Monkey [7] or other automated UI exercisers, to uncover further region-specific behaviors that manifest only during execution.

GeoTwins Dataset. The strict filtering applied in the automated pipeline developed for constructing the GeoTwins dataset may lead to the exclusion of some potential GeoTwins that do not fully meet our predefined criteria. While we performed manual inspection on a statistically significant number of cases to increase confidence, this does not constitute a complete ground truth verification. However, as the GeoTwins dataset represents a novel contribution, we deliberately adopt a highly conservative approach to ensure reliability. Our goal is to enable a scalable, reproducible methodology for studying GeoTwins rather than constructing a perfect ground truth. This design prioritizes precision and minimizes false positives, even at the cost of potentially overlooking some legitimate GeoTwins.

7 Related Work

In this section, we review prior research on geographic variation in mobile applications.

Kumar et al. [27] conducted a large-scale study of Android apps distributed across 26 countries, analyzing approximately 5000 apps. Their work mainly focused on *GeoBlocking*, i.e., when apps are unavailable in certain countries, and the motivations behind this phenomenon, such as government-requested takedowns or developer blocking. They also showed that apps with the same package name can vary in permissions and privacy policies depending on the region from which they are downloaded. More recently, Guo et al. [22] introduced FREELENS, a framework for identifying geographic feature differences (GFDs) at the code level. By analyzing over 21 000 apps across ten countries, they uncovered significant regional disparities in areas such as advertising, authentication, and data handling. However, their study exclusively examined apps with the same package name and explicitly acknowledged the absence of *GeoTwins* in their analysis. For example, in their limitations section, they referenced the GeoTwin pair `com.americanexpress.android.acctsvcs.uk` and `com.americanexpress.`
but did not analyze it, whereas our analysis includes this pair. In addition to covering an unexplored aspect like *GeoTwins* ,apps with similar branding but different package names for targeted regions (see RQ4), we also addressed another limitation common to previous studies. Indeed, we took into account the Android App Bundle format by analyzing complete app bundles and isolating the `base.apk`, thus avoiding expected differences that could be present in the split files (see RQ3).

Various sources have also highlighted the cultural and economic dimensions of regional differences in mobile app ecosystems, discussing how app markets reflect local preferences, platform policies, and socio-economic conditions [2, 25, 31, 36, 42]. Our findings contribute to this discussion by providing empirical evidence of regional differentiation in mobile apps. We observed distinct preferences in third-party libraries (e.g., particularly in Japanese apps), and the prevalence of country-exclusive apps suggests that cultural and economic factors play a significant role in shaping mobile ecosystems. Future work could build on our analysis by incorporating user reviews and ratings to better understand regional user expectations and sentiments.

8 Conclusion

This paper investigated how Android applications vary geographically, moving beyond surface-level accessibility to reveal deeper compositional differences. We introduced and analyzed two previously unexplored phenomena: *GeoTwins*, functionally similar apps distributed under distinct package names for different regions, and unexpected regional variability within supposedly consistent `base.apk` files (e.g., permissions and third-party libraries).

Our analysis shows that *GeoTwins*, despite shared branding, can differ significantly in key aspects such as smali code, files, and URLs. Additionally, the presence of regional variations in `base.apk` files challenges the assumption of a uniform app core across locations, pointing to hidden customizations. These results indicate that location-based differentiation extends to fundamental app structure and behavior.

Overall, our findings highlight that geography shapes not only app availability but also the software itself, raising concerns about privacy equity, platform transparency, and user experience. Future work could expand this study to *GeoFamilies*, groups of regional variants of the same app, and include analyses of user reviews and ratings to better capture regional expectations and sentiments.

9 Acknowledgement

This research was funded in part by the Luxembourg National Research Fund (FNR), REPROCESS grant reference C21/IS/16344458, and UNLOCK grant reference C23/IS/18154263.

References

[1] Marco Alecci, Pedro Jesús Ruiz Jiménez, Kevin Allix, Tegawendé F Bissyandé, and Jacques Klein. 2024. AndroZoo: A Retrospective with a Glimpse into the Future.

In *Proceedings of the 21st International Conference on Mining Software Repositories*. 389–393.
[2] AlgoAce. 2024. *The Importance of Cultural Localization: How Top-Rated Mobile App Development Agencies Adapt for Global Markets*. https://algoace.com
[3] Kevin Allix, Tegawendé F. Bissyandé, Jacques Klein, and Yves Le Traon. 2016. AndroZoo: Collecting Millions of Android Apps for the Research Community. In *Proceedings of the 13th International Conference on Mining Software Repositories* (Austin, Texas) *(MSR '16)*. ACM, New York, NY, USA, 468–471. https://doi.org/10.1145/2901739.2903508
[4] Androguard. [n. d.]. Androguard: A reverse engineering tool for Android applications. https://github.com/androguard/androguard. Accessed: 2025-05-15.
[5] Android Developers. [n. d.]. Guide to Android App Bundles. https://developer.android.com/guide/app-bundle. Accessed: 2025-05-15.
[6] Android Developers. 2024. Manifest.permission | Android Developers. https://developer.android.com/reference/android/Manifest.permission Accessed: 2025-05-27.
[7] Android Developers. n.d.. UI/Application Exerciser Monkey. https://developer.android.com/studio/test/other-testing-tools/monkey.
[8] ApkTool. [n. d.]. ApkTool: A tool for reverse engineering Android APK files. https://ibotpeaches.github.io/Apktool/. Accessed: 2025-05-15.
[9] Appknox. [n. d.]. googleplay-api. https://github.com/appknox/googleplay-api. Accessed: 2025-05-27.
[10] Brasil. 2018. Lei Geral de Proteção de Dados Pessoais (LGPD), Lei No. 13.709/2018. https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/L13709.htm.
[11] James Burroughs, Ashwin Mathew, and Elisa Oreglia. 2025. Using archival versions of apps to understand emerging digital ecosystems. *Big Data & Society* 12, 4 (2025), 20539517251382825.
[12] Haipeng Cai. 2020. Embracing mobile app evolution via continuous ecosystem mining and characterization. In *Proceedings of the IEEE/ACM 7th International Conference on Mobile Software Engineering and Systems*. 31–35.
[13] Haipeng Cai and Barbara Ryder. 2020. A longitudinal study of application structure and behaviors in android. *IEEE Transactions on Software Engineering* 47, 12 (2020), 2934–2955.
[14] Lingwei Chen, Shifu Hou, and Yanfang Ye. 2017. Securedroid: Enhancing security of machine learning-based detection against adversarial android malware attacks. In *Proceedings of the 33rd Annual Computer Security Applications Conference*. 362–372.
[15] Xiao Chen, Chaoran Li, Derui Wang, Sheng Wen, Jun Zhang, Surya Nepal, Yang Xiang, and Kui Ren. 2019. Android HIV: A study of repackaging malware for evading machine-learning detection. *IEEE Transactions on Information Forensics and Security* 15 (2019), 987–1001.
[16] Thomas Cover and Peter Hart. 1967. Nearest neighbor pattern classification. *IEEE transactions on information theory* 13, 1 (1967), 21–27.
[17] DeepL. [n. d.]. DeepL Pro API: Advanced Translation for Developers. https://www.deepl.com/en/pro-api. Accessed: 2025-05-15.
[18] Estados Unidos Mexicanos. 2010. Ley Federal de Protección de Datos Personales en Posesión de los Particulares. https://www.diputados.gob.mx/LeyesBiblio/pdf/LFPDPPP.pdf.
[19] European Commission. [n. d.]. Data Protection. https://commission.europa.eu/law/law-topic/data-protection_en. Accessed: 2025-05-28.
[20] Jun Gao, Li Li, Pingfan Kong, Tegawendé F Bissyandé, and Jacques Klein. 2019. Understanding the evolution of android app vulnerabilities. *IEEE Transactions on Reliability* 70, 1 (2019), 212–230.
[21] google-play-scraper. [n. d.]. google-play-scraper. https://pypi.org/project/google-play-scraper/. Accessed: 2025-05-27.
[22] Jiawei Guo, Yu Nong, Zhiqiang Lin, and Haipeng Cai. 2025. Code Speaks Louder: Exploring Security and Privacy Relevant Regional Variations in Mobile Applications. In *2025 IEEE Symposium on Security and Privacy (SP)*. IEEE Computer Society, 3952–3970.
[23] Richard W Hamming. 1950. Error detecting and error correcting codes. *The Bell system technical journal* 29, 2 (1950), 147–160.
[24] Paul Jaccard. 1901. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bull Soc Vaudoise Sci Nat* 37 (1901), 547–579.
[25] Dmitry Kravtsov. 2019. *Region-specific Design*. https://belitsoft.com/design-for-different-regions Accessed: 2025-05-27.
[26] Neal Krawetz. 2013. Kind of Like That. https://www.hackerfactor.com/blog/?/archives/529-Kind-of-Like-That.html. accessed: 2025-05-30.
[27] Renuka Kumar, Apurva Virkud, Ram Sundara Raman, Atul Prakash, and Roya Ensafi. 2022. A large-scale investigation into geodifferences in mobile apps. In *31st USENIX Security Symposium (USENIX Security 22)*. 1203–1220.
[28] Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, Vol. 10. Soviet Union, 707–710.
[29] Deguang Li, Bing Guo, Yan Shen, Junke Li, and Yanhui Huang. 2017. The evolution of open-source mobile applications: An empirical study. *Journal of software: Evolution and process* 29, 7 (2017), e1855.
[30] Li Li, Tegawendé F Bissyandé, and Jacques Klein. 2017. Simidroid: Identifying and explaining similarities in android apps. In *2017 IEEE Trustcom/BigDataSE/ICESS*. IEEE, 136–143.
[31] Tong Li, Yong Li, Tong Xia, and Pan Hui. 2021. Finding spatiotemporal patterns of mobile application usage. *IEEE Transactions on Network Science and Engineering* (2021).
[32] Wen Li, Xiaoqin Fu, and Haipeng Cai. 2021. Androct: ten years of app call traces in android. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*. IEEE, 570–574.
[33] Tyler McDonnell, Baishakhi Ray, and Miryung Kim. 2013. An empirical study of api stability and adoption in the android ecosystem. In *2013 IEEE International Conference on Software Maintenance*. IEEE, 70–79.
[34] Nippon.com. 2025. *Line and YouTube Most Popular Social Media in Japan*. https://www.nippon.com/en/japan-data/h02384/
[35] OpenAI. [n. d.]. text-embedding-3-small: OpenAI Embedding Model Documentation. https://platform.openai.com/docs/models/text-embedding-3-small. Accessed: 2025-05-15.
[36] Ella Peltonen, Eemil Lagerspetz, Jonatan Hamberg, Abhinav Mehrotra, Mirco Musolesi, Petteri Nurmi, and Sasu Tarkoma. 2018. The hidden image of mobile apps: geographic, demographic, and cultural factors in mobile usage. In *Proceedings of the 20th International Conference on Human-Computer Interaction with Mobile Devices and Services*. 1–12.
[37] Feargus Pendlebury, Fabio Pierazzi, Roberto Jordaney, Johannes Kinder, and Lorenzo Cavallaro. 2019. {TESSERACT}: Eliminating experimental bias in malware classification across space and time. In *28th USENIX security symposium (USENIX Security 19)*. 729–746.
[38] Yohei Sekikawa, Masafumi Kunishige, Taichi Hitomi, and Kazumi Kikuchi. 2025. Exploring the Effect of Social Media and Group Chat Use on Social Isolation Among the Older Adults: A Study in Urban Japan. *Geriatrics* 10, 5 (2025), 131.
[39] Lina F Soualmia, Elise Prieur-Gaston, Zied Moalla, Thierry Lecroq, and Stéfan J Darmoni. 2012. Matching health information seekers' queries to medical terms. *BMC bioinformatics* 13 (2012), 1–15.
[40] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[41] Vultr. [n. d.]. Vultr: Cloud Hosting Made Simple. https://www.vultr.com/. Accessed: 2025-05-27.
[42] Liu Wang, Conghui Zheng, Haoyu Wang, Xiapu Luo, Gareth Tyson, Yi Wang, and Shangguang Wang. 2024. Global Prosperity or Local Monopoly? Understanding the Geography of App Popularity. In *Proceedings of the 21st International Conference on Mining Software Repositories*. 322–334.
[43] Xuetao Wei, Lorenzo Gomez, Iulian Neamtiu, and Michalis Faloutsos. 2012. Permission evolution in the android ecosystem. In *Proceedings of the 28th Annual Computer Security Applications Conference*. 31–40.
[44] Atsuko Yamaguchi, Yasunori Yamamoto, Jin-Dong Kim, Toshihisa Takagi, and Akinori Yonezawa. 2012. Discriminative application of string similarity methods to chemical and non-chemical names for biomedical abbreviation clustering. In *BMC genomics*, Vol. 13. Springer, 1–10.
[45] Hongwei Zhang, Yongkang Peng, Shuhong Liao, and Jingrong Mao. 2023. A Method for Protecting Ceramic Pattern Property Rights Based on ResNet50. In *Proceedings of the 2023 4th International Conference on Computer Science and Management Technology*. 345–348.
[46] Jack Zhang, Shikhar Sagar, and Emad Shihab. 2013. The evolution of mobile apps: An exploratory study. In *Proceedings of the 2013 International Workshop on Software Development Lifecycle for Mobile*. 1–8.
[47] Jingjing Zheng, Shuhua Teng, Peirong Li, Wei Ou, Donghao Zhou, and Jun Ye. 2021. A novel video copyright protection scheme based on blockchain and double watermarking. *Security and communication networks* 2021, 1 (2021), 6493306.