Noname manuscript No.
(will be inserted by the editor)

An Empirical Study of the Characteristics of Popular Minecraft Mods

Daniel Lee · Gopi Krishnan Rajbahadur · Dayi Lin · Mohammed Sayagh · Cor-Paul Bezemer · Ahmed E. Hassan

Received: date / Accepted: date

Abstract It is becoming increasingly difficult for game developers to manage the cost of developing a game, while meeting the high expectations of gamers. One way to balance the increasing gamer expectation and development stress is to build an active modding community around the game. There exist several examples of games with an extremely active and successful modding community, with the Minecraft game being one of the most notable ones.

This paper reports on an empirical study of 1,114 popular and 1,114 unpopular Minecraft mods from the CurseForge mod distribution platform, one of the largest distribution platforms for Minecraft mods. We analyzed the relationship between 33 features across 5 dimensions of mod characteristics and the popularity of mods (i.e., mod category, mod documentation, environmental context of the mod, remuneration for the mod, and community contribution for the mod), to understand the characteristics of popular Minecraft mods. We firstly verify that the studied dimensions have significant explanatory power in distinguishing the popularity of the studied mods. Then we evaluated the contribution of each of the 33 features across the 5 dimensions. We observed that popular mods tend to have a high quality description and promote community contribution. In addition, simplifying the mod development is positively correlated with mod popularity.

Keywords mods · mod development · CurseForge · Minecraft

Daniel Lee · Gopi Krishnan Rajbahadur · Dayi Lin · Mohammed Sayagh · Ahmed E. Hassan
Software Analysis and Intelligence Lab (SAIL)
Queen's University
Kingston, ON, Canada
E-mail: {dlee, krishnan, dayi.lin, msayagh, ahmed}@cs.queensu.ca

Cor-Paul Bezemer
Analytics of Software, Games and Repository Data (ASGAARD) Lab
University of Alberta
Edmonton, AB, Canada
E-mail: bezemer@ualberta.ca

1 Introduction

The team size, cost and complexity in game development can grow exponentially as the user requirements increase [84]. Thus, it has become challenging to develop a successful game, and game developers are constantly under an immense amount of stress [72].

One approach to balance the increasing gamer expectation and development stress is to build an active modding community around the game. Skyrim and Minecraft are examples of games that have been successful in building active modding communities [34, 97] to increase the longevity of the games. For example, the Skyrim game still has a median of 86 new mods released per day 8 years after its initial game release in 2011, along with more than 514M total unique downloads of mods [81]. Prior work also shows that an active modding community can contribute to the increased sales of the original game [74].

There are two key components of an active modding community of a game: the active development of mods, and the active adoption of mods by gamers. In our prior work, we looked at how game developers can help maintain the active development of mods, and observed that games from developers with a consistent modding support within the same or different game franchises, were associated with faster releases of mods [45]. In this paper, we identify the characteristics that distinguish popular mods from unpopular ones. To do so, we study 33 characteristics along 5 dimensions of 1,114 popular and 1,114 unpopular mods for the Minecraft game from the CurseForge mod distribution platform – one of the largest distribution platforms for Minecraft mods. We focus on the mods from the Minecraft game because it has one of the largest and most active modding communities [57]. In particular, we answer the following two research questions (RQs):

RQ1: Do our studied dimensions have enough explanatory power to distinguish popular mods from unpopular ones?

Motivation: The goal of this research question is to investigate how well each studied dimension of characteristics (i.e., features) of mods can individually distinguish the popular mods from unpopular ones. We also investigate how well all the studied dimensions together can distinguish popular mods from unpopular ones. Prior work [92] used similar dimensions to identify the characteristics that distinguish mobile apps with high ratings from the ones with low ratings. The results of this research question lay the foundation for further investigations of the characteristics of popular mods.

Findings: We observed that each studied dimension of characteristics of a mod has significant explanatory power in distinguishing popular from unpopular mods. Among the studied dimensions, the community contribution for the mod dimension has the largest explanatory power. However, our combined model which uses all the features across the five dimensions outperforms the best model using an individual dimension by 10% (median).

RQ2: What features best characterize a popular mod?

Motivation: The goal of this research question is to investigate which features of mods can best characterize popular mods. The results of RQ1 show that the

Table 1: An overview of Minecraft mod distribution platforms

Minecraft mod distribution platform	**# of mods**
CurseForge[1]	12,710
Planet Minecraft[2]	9,159
Minecraft Six[3]	3,880
Minecraft Mods[4]	532

[1] https://minecraft.curseforge.com
[2] https://www.planetminecraft.com
[3] http://minecraftsix.com
[4] https://www.minecraftmods.com

studied features have a strong explanatory power for the popularity of a mod. In this RQ, we further investigate the characteristics of popular mods at a granular level.
Findings: We observed that 18 of the 33 (54.5%) studied features help in distinguishing popular mods from unpopular ones. Simplifying the mod development is positively correlated with mod popularity. In addition, popular mods tend to promote community contribution with a source code repository URL and an issue tracking URL, and have a richer mod description.

The remainder of the paper is outlined as follows. Section 2 gives background information about the Minecraft game and the CurseForge mod distribution platform. Section 3 gives an overview of related work. Section 4 discusses our methodology. Sections 5 discusses the results of our empirical study. Section 6 outlines threats to the validity of our findings. Section 7 concludes our study.

2 Background

This section provides a brief overview of the Minecraft game and the CurseForge mod distribution platform.

2.1 The Minecraft Game

The Minecraft game is an open-ended 3D sandbox game, initially developed in the Java programming language, where gamers can use various resources (e.g., blocks) to create their own worlds [58]. Developed by the Mojang[1] game studio, the Minecraft game is one of the best selling video games of all time in 2019, with over 176 million copies sold since its release in 2011 [13]. Mods are considered one of the most popular aspects of the Minecraft game, and are credited for the great success of the game [28, 30, 67].

[1] https://mojang.com/

2.2 The CurseForge Mod Distribution Platform

Minecraft mods on CurseForge. The CurseForge mod distribution platform hosts one of the largest online Minecraft mod repositories with more than 12,000 downloadable mods [23]. Table 1 shows a comparison of the CurseForge mod distribution platform to other Minecraft mod distribution platforms with respect to the number of mods. The CurseForge mod distribution platform provides a dedicated page for each mod. The dedicated page contains detailed information about a mod including contributors, releases, and dependencies, while categorizing the mod under at least one mod category. Furthermore, mod developers can provide their Paypal[2] or Patreon[3] donation URLs on their mod's page. Patreon is a crowdfunding platform where content creators such as mod developers can promote themselves, and receive monthly donations.

Mod contributors on CurseForge. A mod on the CurseForge mod distribution platform can have multiple contributors, and each contributor is assigned a role for the mod (i.e., artist, author, contributor, documenter, former author, maintainer, mascot, owner, tester, ticket manager, or translator). There can be multiple contributors of a mod with the same role, except for the "owner" role which is only assigned to the user that creates the mod on the platform. Unfortunately, the CurseForge mod distribution platform does not provide any official definition for the roles. Furthermore, we observed that the number of mod developers in a mod does not always accurately represent the actual number of contributors. For example, the *Fossils and Archeology Revival* mod[4] shows 10 mod developers on the CurseForge page, but the mod has 17 contributors on Github. Hence, we do not use the mod developer roles or the number of mod developers in our study.

Mod releases and dependencies on CurseForge. The dedicated page of each mod on the CurseForge mod distribution platform lists the mod releases with corresponding upload dates and supported Minecraft, Java, and Bukkit[5] versions. In addition, the dependencies for each release are also listed on a mod's page. The CurseForge mod distribution platform supports the declaration of several types of dependencies of a mod release, including "incompatible", "tool", "required", "embedded library", and "optional dependencies".

3 Related Work

This section discusses prior studies that are related to our study. We discuss related work on (1) empirical studies of game mods, (2) games and software engineering, (3) studies of the Minecraft game, and (4) mining online software distribution platforms.

[2] https://www.paypal.com/

[3] https://www.patreon.com/

[4] https://minecraft.curseforge.com/projects/fossils

[5] Bukkit is a Minecraft Server mod that helps in the running and modification of a Minecraft server. See https://bukkit.org/pages/about-us/ for more details.

3.1 Empirical Studies of Game Mods

Several prior studies studied the modding community to identify and analyze the relationship between mod developers and the game industry, yielding insights on collaborative practices and strategies, as well as capturing the value of mods [5, 40, 66]. A few prior studies mined data from the Nexus Mods distribution platform to quantitatively study the motivation behind mod developers based on the users' expectations, and to understand how to build and maintain an active modding community [24, 45]. Particularly, Dey et al. [24] study the meta data available for popular and unpopular mods of six famous PC games across several popular online mod distribution platforms to investigate the motivations of mod developers. They find that user demands and the content created by the mod developers correlate very weakly and suggest that more effort needs to undertaken to bridge this gap. Furthermore, similar to our study they also seek to investigate what features make a mod popular. However, they consider only the general tags associated with a given mod and they do it across multiple games without any consideration to the game-specific characteristics.

Additionally, Poretski and Arazy [74] conducted an empirical study on 45 games from the Nexus Mods distribution platform and observed that mods increased the sales of the original game. Targett et al. [90] empirically studied user-interface mods of the World of Warcraft[6] game to gather insights on how mods contribute to the World of Warcraft game and its modding community. They observed that modifications helped the interface of video games meet the needs of users, since every user has their own ideal interface.

Similarly, Wu et al. [95] studied popular Reddit threads on Minecraft mod discussions to uncover the learnt knowledge by Minecraft modders. They assert that these threads contain vast peer-generated knowledge on how to create artifacts in the Minecraft environment. Levitt [44] studied the evolution of the creative process around the creation of Minecraft mods. Additionally, several studies [43, 65] investigated Minecraft mods and their role in enhancing individual creativity and general interest in the field of Science, Technology, Engineering and Mathematics (STEM). They found that modding in the context of the Minecraft game positively influenced both of these aforementioned aspects. Beggs [11] studied how the dynamics between producers and consumers within the game industry are impacted by modding. They did so by studying Minecraft mods. Beggs observed that Minecraft modders in total spend close to 3 million hours weekly creating and maintaining mods. Furthermore, they also noted that the modding culture pushes game consumers into generally preferring games that allow modding.

Different from the aforementioned studies, we study the characteristics that distinguish popular mods from unpopular ones specific to a particular game (Minecraft) in order to better understand the characteristics of popular mods.

[6] https://worldofwarcraft.com/en-us/

3.2 Games and Software Engineering

Several studies investigated open source game projects to relate them to software engineering aspects [1, 68]. For instance, Pascerella et al.[68] investigated how the developers contribute to video games in an open source setting. A few studies analyzed the development of the authors' own video games [31, 42], while Guana et al. [32] studied the development of their own game engine. In particular, Guana et al. [32] outline how game development is more complicated than traditional software development and presents a model-driven approach to simplify the development of game engines. Bécares et al. [10] investigated the gameplay of the *Time and Space* game and outlined an approach to automate the game tests.

A few prior studies studied the videos of game-related bugs [48]. Notably, Lin et al. [51] identified gameplay videos that showcase game bugs, as naïve methods such as keyword search is inaccurate. They proposed a random forest classifier that outperforms other classifiers (i.e., logistic regression and neural network), and provides a precision that is 43% higher than the naïve keyword search approach. Furthermore, several studies [47, 73, 93] have been conducted on the postmortems of games based on articles/magazines to draw insights on the do's and dont's of game development.

Ampatzoglou and Stamelos [4] provided researchers with a systemic review on available literature. In addition, Scacchi and Cooper [80] extensively analyzed the software engineering literature of games.

Rather than investigating the software engineering aspect of the original game, in this paper we conduct an empirical study by mining the software engineering aspects of game mods that are available in the CurseForge platform.

3.3 Studies of the Minecraft Game

Several prior studies have examined the Minecraft game for pedagogical uses [3, 8, 9, 18, 25, 26, 35, 46, 64, 71, 83, 86, 87, 97]. In addition, Nebel et al. [64] conducted an extensive literature review on the usage of the Minecraft game in education. A few prior studies primarily focused on using the Minecraft game to study the players of the game [19, 61, 76]. Furthermore, a few prior studies primarily focused on using the Minecraft game to streamline the development of software [6, 79].

In our study, we analyze Minecraft mods to provide an empirical understanding of the characteristics of popular mods.

3.4 Mining Online Software Distribution Platforms

Mining online software distribution platforms to provide useful information and insights about the popularity of software has been a fundamental part of software engineering research. We present a brief summary of how mining online software distribution platforms has been carried out in the context of traditional software, games and mobile apps.

Traditional software. GitHub is one of the most popular online code hosting distribution platforms for traditional software. Several prior studies investigated the popularity of software projects in GitHub to provide insights to software developers [14–17, 41, 96]. For example, Borges et al. [17] outline how a GitHub repository gathers popularity over time. In addition, Borges et al. outline the characteristics of successful GitHub repositories for other software developers to mimic. Similarly, Zhu et al. [96] suggest that better folder organizational practices lead to better project popularity in GitHub.

Mobile apps. Many prior studies investigated features that impact the success of a mobile app by mining data from mobile app stores to provide useful guidelines to mobile app developers [7, 21, 33, 53, 88, 92]. For example, Tian et al. [92] studied the differences between popular and unpopular mobile apps and found that popular apps generally have more complex code and better exploit the latest features of the target Android SDK (Software Development Kit). Taba et al. [88] studied how the complexity of the UI of a mobile app affects its popularity and provided guidelines to developers on the amount of UI complexity they should strive for in order to keep their users happy. Similarly, Bavota et al. [7] and Linares-Vásquez et al. [53] studied the characteristics of the APIs used by popular and unpopular apps and recommended developers to use less defect-prone and change-prone APIs to ensure the popularity of their mobile apps.

Games. Prior studies that mine data from online game distribution platforms primarily focused on extrapolating useful insights for game developers from platforms such as Steam [12, 52, 85]. For example, Lin et al. [49] studied urgent updates on the Steam platform and observed several update patterns to help developers avoid undesirable updates. Lin et al. [50] also studied the early access model on the Steam platform and suggested that game developers use the early access model to elicit early feedback and gather more positive feedback. Cheung et al. [20] investigated over 200 Xbox 360 game reviews to understand how the first hour of gameplay engages new players. Similarly, Ahn et al. [2] analyzed game reviews between popular and unpopular games on the Steam platform to better understand the characteristics of popular Steam games, and offered guidance to game developers on how to make their game popular.

Though many studies mined various software repositories and provided insights to developers, these insights do not directly translate to mod developers as software such as mobile apps and games are developed from the ground-up for the consumption of users. In contrast, game mods are software that was built to enhance, extend or provide (new) features to an existing game in a meaningful way by hacking the source code of the original or through official APIs. Several prior studies [62, 68–70] show that video game development is starkly different from other types of software development. Therefore, by extension, we expect game mod development (which is a subset of game development) to be different from mobile app and video games development. For instance, consider these two studies by Tian et al. [92] and Ahn et al. [2]. Both studies examine the characteristics of popular mobile apps and video games by mining the Google Play store and the Steam platform respectively to provide insights to mobile app and video game developers. For the mobile app developers, Tian et al. [92] suggest that size of the app, number of promotional images and the

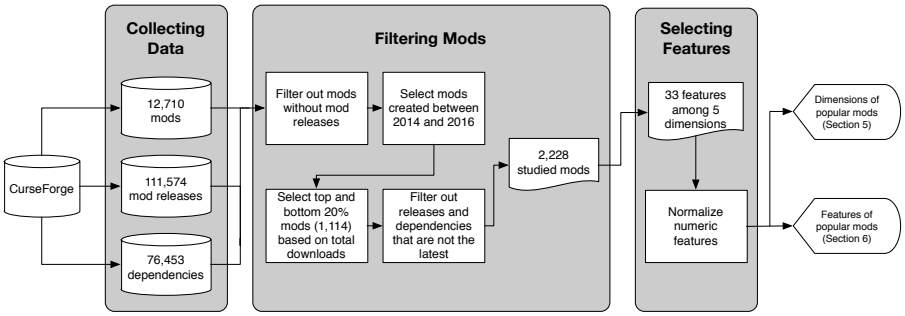


Fig. 1: An overview of our data collection.

target SDK are the three key elements that are associated with the popularity of a mobile app. In contrast, Ahn et al. [2] recommend developers to improve the gameplay, the challenge and the motivational aspects and emotional connect of the video game while lowering the price and improving the games storyline. However, different from both of these studies, from studying the CurseForge platform we find that popular mods are likely to have a better mod description, ease other mod development and welcome community contributions. Such a result further signifies that game mods are different from other types of software.

Hence, the findings and recommendations for mobile developers, game developers and traditional software developers to ensure the popularity of their software as prescribed by prior studies cannot be directly transferred to game mod developers. Therefore, a study such as ours is pivotal in understanding the characteristics of popular mods. We envision future studies to build on our work in order to help developers improve the popularity of their mods.

We did however conduct our study in the same vein as the aforementioned studies by mining the CurseForge mod distribution platform to gain an empirical understanding of the characteristics of popular mods. To the best of our knowledge, the study by Dey et al. [24] is the only other study that mines online mod distribution platforms to study the characteristics of popular mods. However, they focus only on the tags that are provided for the mods on the distribution platforms and do not endeavour to provide insights to mod developers.

We study the characteristics of popular and unpopular mods specific to a particular game (Minecraft) to better understand what characterizes popular mods. These characteristics can be further explored by future work to assist mod developers in improving the quality of their mods. Furthermore, we are the first to conduct a statistically rigorous analysis on 33 features collected across 5 dimensions to generate insights for mod developers.

Table 2: An overview of the CurseForge mod distribution platform dataset.

Number of total mods	12,710
Number of studied mods	2,228
Number of studied dimensions	5
Number of studied features	33
Number of total mod releases	111,574
Number of total mod dependencies	76,453

4 Methodology

This section discusses the methodology of our empirical study of the characteristics of popular and unpopular Minecraft mods. Figure 1 gives an overview of our methodology.

4.1 Collecting Data

We collected the dataset for our study from the CurseForge mod distribution platform on June 6, 2019, using a customized crawler. Table 2 shows an overview our Minecraft mod dataset.

Collecting Mods. We collected the information of 12,710 mods. In particular, we collected the name, categories, number of total comments, source code URL, issue tracking URL, Paypal URL, and Patreon URL for each mod.

Collecting Mod Releases. We collected the information of 111,574 releases across all mods. In particular, we collected the type, upload date, size, number of downloads, and supported Minecraft, Java, and Bukkit versions for each mod release.

Collecting Dependencies. We collected 76,453 mod dependencies across all mod releases. In particular, we collected the type, mods, and the direction for each dependency.

4.2 Filtering Mods

To ensure the quality of the studied mods, we first removed 295 inactive mods that have no mod releases. Then, we removed 6,845 mods that were created before 2014 or after 2016 to ensure the studied mods all have an equal chance to obtain a high number of downloads. For the remaining 5,570 mods, we selected the top and bottom 20% of the mods based on their total number of downloads for our study. We consider the top 20% of mods (1,114 mods) as popular mods, and the bottom 20% of mods (1,114 mods) as unpopular mods based on their total number of downloads. Hence the claims that are made about a mod being (un)popular are about the likelihood of the mod belonging to the most/least popular group of mods.

We do not take into account the lifetime of a mod (despite some mods being created in 2014 and some mods being created in 2016) when separating the mods into

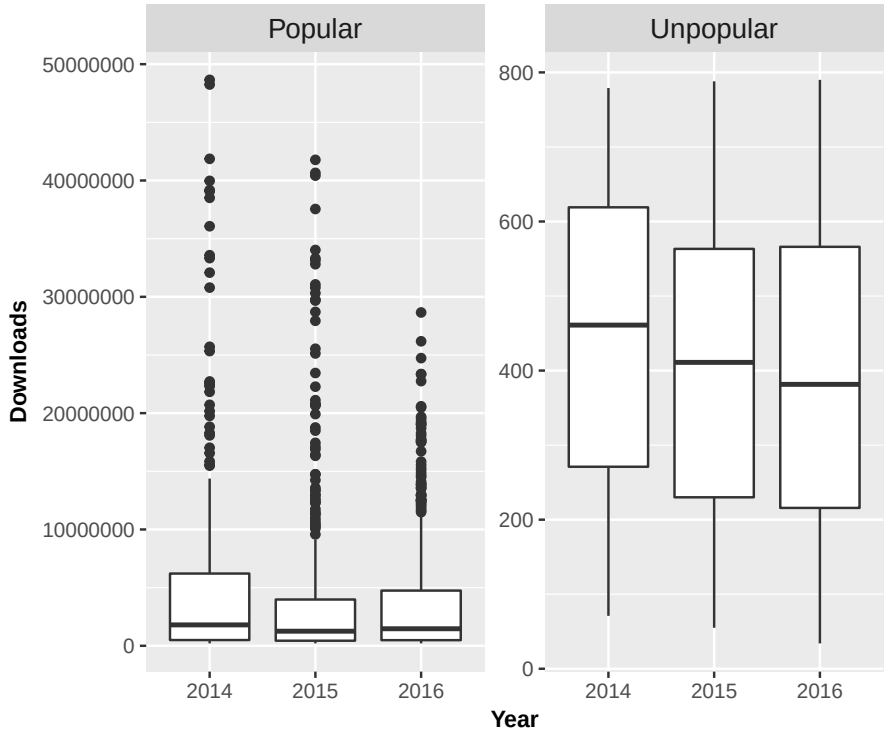


Fig. 2: Distribution of the number of downloads that are received by popular and unpopular mods that are created in 2014, 2015 and 2016.

popular and unpopular groups. We do so as the number of median downloads across the studied years for mods in the popular and unpopular groups remains relatively consistent as we can observe from Figure 2. Furthermore, we observed that the number of popular mods that were created each year in the studied period also remains consistent. More specifically, among the 1,114 popular mods, 279 were created in 2014, and 415 and 418 mods were created in 2015 and 2016 respectively. In total, we studied 2,228 mods. Our selection approach is similar to prior study [92] which selected the highest and lowest rated mobile apps for study.

We choose to study the number of downloads as a proxy for the popularity of a mod, as this number acts as a good indicator of the needs for the provided features/alterations by the mod within the Minecraft community. Furthermore, a mod becoming popular in an online platform like CurseForge is pivotal for the mod developers. For instance, as Postigo et al. [75] outline, mod developers want their mods to be popular as being known in the modding community may open up potentially lucrative job opportunities. Finally, identifying features that affect the popularity of software in online distribution platforms is widely regarded as an important software engineering challenge [63]. This importance is for example demonstrated by the many software

engineering studies that examine the characteristics of popular mobile apps in app stores (e.g.,[7, 36, 53, 92]).

For each of the 2,228 mods, we used the information of the mod's latest release and dependencies in our study.

4.3 Selecting Features

Starting from prior work on the popularity of mobile apps [92] and our own intuition, we defined 5 dimensions that might be associated with the popularity of mods (i.e., mod category, mod documentation, environmental context of the mod, remuneration for the mod, and community contribution for the mod). Then, we define for each dimension the features that are available on the CurseForge platform and that we can extract in an automated fashion. We end up with 33 features (characteristics) that we leverage to understand the differences between the characteristics of popular and unpopular Minecraft mods. Table 3 shows an overview of the 33 features and their associated dimensions, along with their corresponding explanation and rationale. In addition, we normalized all features with the '*numeric*' type in Table 3 using a `log(1 + x)` transformation to reduce the bias caused by the outliers.

5 Characteristics of Popular and Unpopular Minecraft Mods

In this section, we present the results of our empirical study of the characteristics of popular and unpopular Minecraft mods.

5.1 RQ1: Do our studied dimensions have enough explanatory power to distinguish popular mods from unpopular ones?

Motivation: In this research question, we investigate how well each studied dimension of characteristics (i.e., features) of mods can individually distinguish the popular mods from unpopular ones. We also investigate how well can all the studied dimensions together distinguish popular mods from unpopular ones. Prior study [92] used similar dimensions to identify the characteristics that distinguish mobile apps with high ratings from the ones with low ratings. The results of this research question lay the foundation for further investigations of the characteristics of popular mods.
Approach: To investigate how well the individual dimensions can distinguish popular mods from unpopular ones (i.e., their explanatory power), we built a logistic regression model for each dimension in Table 3. We used logistic regression, instead of other complex techniques (e.g., a neural network) as logistic regression is transparent and interpretable [59, 78]. In particular, for each dimension's model, we used the features in a dimension as independent variables and whether the mod is popular as the dependent variable. We consider the given dimension to have significant explanatory power if the AUC of the model constructed with the dimension is greater than 0.5, which means that the dimension can distinguish popular from unpopular mods. The dimension that results in the largest AUC is deemed to have the most explanatory

Table 3: Dimensions and their features describing the characteristics of popular and unpopular Minecraft mods.

Dimension	Feature Name	Explanation	Type	Rationale
Mod Category	Number of categories (*num_categories*)	Total number of categories that a mod belongs to. A mod must belong to at least one category.	Numeric	Mods that offer a variety of categories can attract users with more options.
	Miscellaneous *(is_cat_misc)*	Mods that do not belong to any of the existing categories. For example, the *OpenBlocks* mod.[1]	Boolean	Certain Minecraft mod categories in the CurseForge mod distribution platform may attract more users to the mod.
	Food *(is_cat_food)*	Mods that provide changes to anything related to food in-game. For example, the *AppleSkin* mod.[2]	Boolean	
	World generation *(is_cat_world_gen)*	Mods that provide changes related to the world, such as new terrains.	Boolean	
	Magic *(is_cat_magic)*	Mods that provide changes related to magic in the Minecraft game. For example, the *Roots* mod.[3]	Boolean	
	API and library *(is_cat_library_api)*	Mods that provide shared code for other mod developers to use.	Boolean	
	Fabric *(is_cat_fabric)*	Mods that are created using the *Fabric*[4] modding toolchain.	Boolean	
	Technology *(is_cat_technology)*	Mods that provide changes for any in-game technology.	Boolean	
	Armor, tools, and weapons *(is_cat_armor_weapons_tools)*	Mods that provide changes to in-game armor, weapons, and tools.	Boolean	
	Addons *(is_cat_addons)*	Mods that provide utilities for mod users to easily extend in-game features.	Boolean	
	Adventure and RPG *(is_cat_adventure_rpg)*	Mods that change the gameplay experience of the in-game adventure.	Boolean	
	Server utility *(is_cat_server_utility)*	Mods that provide changes to the server-side of the Minecraft game.	Boolean	
	Redstone *(is_cat_redstone)*	Mods that are provide changes related to the redstone resource in the Minecraft game.	Boolean	
	Map and information *(is_cat_map_info)*	Mods that provide changes related to the location and information on items.	Boolean	
	Storage *(is_cat_storage)*	Mods that provide mod users blocks and items, which improve the existing in-game storage.	Boolean	
	Twitch integration *(is_cat_twitch_integration)*	Mods that provide changes related to the interaction between the mod and the Twitch platform.	Boolean	
	Cosmetic *(is_cat_cosmetic)*	Mods that provide changes to the texture and aesthetic of the in-game models.	Boolean	
Mod Documentation	Number of words in the short description *(num_words_short_desc)*	Number of words in the mod's preview description.	Numeric	The longer the description, the more likely that mod users will understand what the mod offers without downloading the mod.
	Number of words in the long description *(num_words_long_desc)*	Number of words in the mod's main description.	Numeric	
	Mod wiki URL *(is_mod_wiki_url)*	An external link with the documentation of a mod.	Boolean	The presence and quality of a mod's documentation can help other mod users understand how to utilize the mod to its full potential, which can give users a better experience.
	Number of images *(num_images)*	Number of in-game screenshots that a mod has.	Numeric	In-game screenshots can help promote and visually explain the mod's functionalities, which may attract users, without trying the mod first.

Environmental Context of the Mod	Latest number of incompatible dependencies *(num_incompatible_dep)*	Number of incompatible dependencies that are in the latest release of the mod, which means that another mod is not compatible with the mod.	Numeric	Dependencies provide mods more functionalities, which could make the mod appeal to more users.
	Latest number of tool dependencies *(num_tool_dep)*	Number of words in the mod's main description.	Numeric	
	Latest number of required dependencies *(num_required_dep)*	Number of required dependencies that are in the latest release of the mod, which means another mod is required to make the mod function.	Numeric	
	Latest number of embedded library dependencies *(num_embedded_lib_dep)*	Number of embedded library dependencies that are in the latest release of the mod, which provides shared code for the mod's development.	Numeric	
	Latest number of optional dependencies *(num_optional_dep)*	Number of optional dependencies that are in the latest release of the mod, which means the dependency adding a certain functionality can be switched on and off.	Numeric	
	Latest number of supported Minecraft version(s) *(latest_num_mc_versions)*	Number of Minecraft versions supported by the latest mod release, which corresponds to a specific version of the Minecraft game. Mods must support at least one Minecraft version.	Numeric	A larger number of supported versions could attract more users to a mod by providing more stability, and access to functionalities from different versions.
	Latest number of supported Java version(s) *(latest_num_java_versions)*	Number of Java versions supported by the latest mod release. Mod developers can optionally provide this information.	Numeric	
	Latest number of supported Bukkit version(s) *(latest_num_bukkit_versions)*	Number of Bukkit API[5] versions supported by the latest mod release. The Bukkit API extends the multiplayer server of the Minecraft game for others to modify. Mod developers can optionally provide this information	Numeric	
Remuneration for the Mod	PayPal URL *(is_paypal_url)*	An external link to PayPal for donations.	Boolean	Mod developers that ask for donations are more likely to be dedicated to modding, which can attract more users.
	Patreon URL *(is_patreon_urls)*	An external link to Patreon for donations.	Boolean	
Community Contribution for the Mod	Source code URL *(is_mod_source_code)*	An external link to the source code of a mod (e.g., Github).	Boolean	Mods that provide a link to their source code could invite more contributors, which could attract users with more content at a faster speed.
	Issue tracking URL *(is_mod_issues)*	An external link to an issue tracking system.	Boolean	Mods that provide a link to an issue tracking system could indicate to a user that a mod is more stable, which may attract them to the mod.

[1] https://www.curseforge.com/minecraft/mc-mods/openblocks
[2] https://www.curseforge.com/minecraft/mc-mods/appleskin
[3] https://www.curseforge.com/minecraft/mc-mods/roots
[4] http://fabricmc.net/
[5] https://dev.bukkit.org/

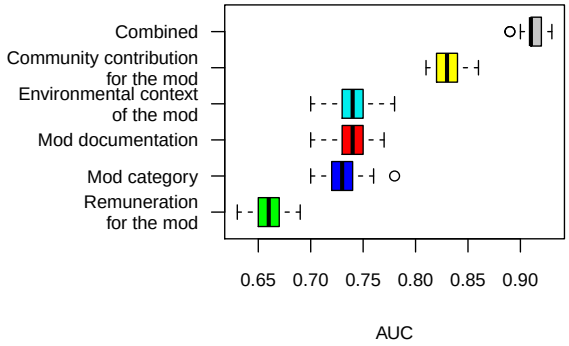


Fig. 3: The distribution of the AUCs of models constructed with an individually studied dimension, and with all studied dimensions combined. The different colors represent the statistically different ranks given by the Scott-Knott effect size difference test. The distributions are sorted by their ranks (presented in ascending order from left to right with remuneration for the mod having the lowest rank) from the Scott-Knott effect size difference test.

power and vice versa. We used the `glm` function[7] from the `stats` package[8] to create the logistic regression models.

To validate the performance of our built models, we performed 100 out-of-sample bootstrap iterations to compute the AUC (Area Under the receiver operator characteristics Curve) for each model. Prior study [89] showed that the out-of-sample bootstrap technique had the best balance between the bias and variance of estimates. The out-of-sample bootstrap technique randomly samples data with replacement for `n` iterations. The sampled data in an iteration is used as the training set for that iteration, while the data that was not sampled in that iteration is used as the testing set for that iteration. We then trained a model with the training set and calculated the AUC of the model with the testing set for each iteration.

In addition, to investigate how well all studied dimensions combined can distinguish popular mods from unpopular mods, we built a logistic regression model using all 33 features from the 5 dimensions in Table 3. We evaluated the performance of this combined model using the same aforementioned process of computing the AUC of the model with 100 out-of-sample bootstrap iterations.

Furthermore, we used the Scott-Knott effect size difference test to statistically sort and rank the distributions of the AUCs of all studied dimensions [89]. We used

[7] https://www.rdocumentation.org/packages/stats/versions/3.6.1/topics/glm
[8] https://www.rdocumentation.org/packages/stats/versions/3.6.1

the `sk_esd` function[9] from the `ScottKnottESD` package[10] for the Scott-Knott effect size difference test.

Findings: Each studied dimension has significant explanatory power to individually identify popular mods. Figure 3 shows the distribution of AUCs per studied dimension. The lowest median AUC among the studied dimensions was 0.66, implying that every dimension has significant explanatory power (i.e., the model has an AUC > 0.5) in distinguishing popular mods from unpopular ones. In addition, the Scott-Knott effect size difference test shows a statistical significant difference between each studied dimensions, with non-negligible effect sizes. Among the studied dimensions, the community contribution for the mod dimension is ranked as having the largest explanatory power, whereas the remuneration for the mod dimension is ranked as having the lowest explanatory power.

The combined model has a larger explanatory power than each of the studied dimension individually. Figure 3 shows the distribution of AUCs of the *combined model* that combines all studied dimensions together. The combined model has the largest median AUC of 0.91, outperforming every one of the studied dimensions on their own. The Scott-Knott effect size difference test confirms that the combined model has the highest ranking in explanatory power compared to the individual studied dimensions.

In addition, Figure 3 shows that the combined model has a 10% higher median AUC than the community contribution for the mod dimension (the dimension with the highest explanatory power among the studied dimensions), and a 38% higher median AUC than the remuneration for the mod dimension (the dimension with the lowest explanatory power among the studied dimensions). Prior study [92] also observed that a combined model with all the dimensions has a larger explanatory power than models with individual dimensions in the context of distinguishing mobile apps with high ratings from mobile apps with low ratings.

> *Each studied dimension of characteristics of a mod has significant explanatory power in distinguishing popular from unpopular mods. Among the studied dimensions, the community contribution for the mod dimension has the largest explanatory power. However, our combined model which uses all the features across the five dimensions outperforms the best model using individual dimension by 10% (median).*

5.2 RQ2: Which features best characterize a popular mod?

Motivation: In this research question, we investigate which mod features can best characterize popular mods. The results of RQ1 show that the studied dimensions have a strong explanatory power for the popularity of a mod. In this RQ, we further investigate the characteristics of popular mods at the feature-level across 33 features and

[9] https://www.rdocumentation.org/packages/ScottKnottESD/versions/1.2.2/topics/%22sk_esd%22

[10] https://www.rdocumentation.org/packages/ScottKnottESD/versions/1.2.2

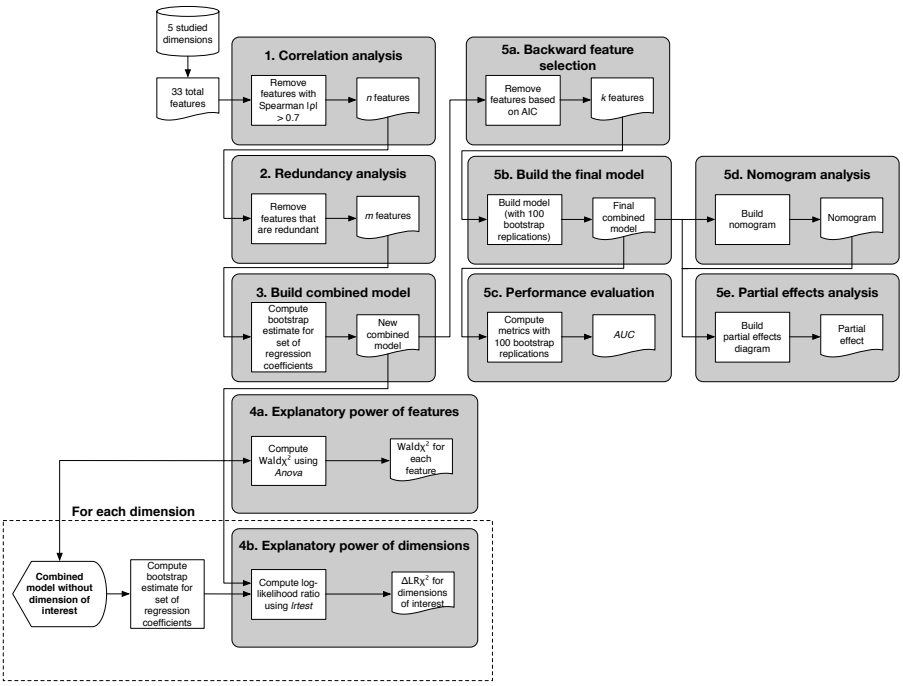


Fig. 4: An overview of the process that we used to build, evaluate and analyze the combined model.

5 dimensions to systematically quantify the association between the studied features and the number of downloads for a mod.

Approach: To investigate which features can best characterize popular mods, in this research question we focus on analyzing the combined model with all dimensions of features, as RQ1 shows that the combined model has the most explanatory power for mod popularity.

Figure 4 shows an overview of our approach to construct, evaluate and analyze the combined model. Below we explain each step in detail:

1. **Correlation analysis.** We performed correlation analysis to reduce collinearity between the features before we built the models, since correlated features can affect the interpretation of the model [55, 56]. We used the `varclus` function[11] from the `Hmisc` package[12] in `R` to filter out highly correlated features. We calculated *Spearman's correlation coefficients* among the studied features. We consider a pair of features with a *Spearman correlation coefficient* $>= 0.7$ as highly correlated. We did not observe high correlations among our studied features.
2. **Redundancy analysis.** Before building the models, we also performed redundancy analysis to eliminate redundant features that can interfere with the relation-

[11] https://www.rdocumentation.org/packages/Hmisc/versions/4.2-0/topics/varclus
[12] https://cran.r-project.org/web/packages/Hmisc/index.html

ship between the independent variables (i.e., features), which in turn may distort the relationship the independent variables have with the dependent variable (i.e., popularity) [55]. We used the `redun` function[13] from the `Hmisc` package in `R` to filter out features that can be linearly predicted by other features. We removed the '*number of categories*' feature as it is redundant, leaving 32 features for the remainder of the study.

3. **Building the combined model.** We used all the remaining features after step 2 to build a logistic regression model. However, the model's regression coefficients could vary or be estimated incorrectly based on the sample of data and the underlying assumptions [29]. Hence, to avoid biasing the estimated regression coefficients, we used the `bootcov` function from the `rms` package using 100 bootstrap iterations to adjust the regression coefficients with bootstrap estimates, to ensure the non-arbitrariness of the estimated regressions co-efficients in the combined model [38, 39].

4a. **Explanatory power of features.** We used Wald's χ^2 to measure the explanatory power of the features in the model from step 3. The larger the Wald χ^2, the larger the explanatory power of the feature [39]. Prior study [91] used the same approach to compute the explanatory power of features. We computed the Wald χ^2 with the `Anova` function[14] from the `car` package[15] in `R` using the parameter *test.statistic='Wald'*. Table 4 shows the explanatory power of each feature (Wald χ^2).

4b. **Explanatory power of dimensions.** Though in RQ1, we observed that each dimension of features of a mod has explanatory power, we are uncertain of the unique explanatory power each of them contains in relation to the other dimensions. Understanding the unique explanatory power of each dimension is critical to assert which of these dimensions matter the most for characterizing the popularity of a mod. For example, from Figure 3 we observe that the environmental context of the mod and mod documentation dimensions by themselves can explain the popularity of a mod with a median AUC of 0.74. However, we are uncertain of how much unique power each of these dimensions contribute to the model built on all the studied dimensions, which had a median AUC of 0.92.

Therefore, we conducted a chunk test on each of the studied dimensions in the combined model from step 3, to quantify the explanatory power of each studied dimension [37, 55]. For each of the studied dimensions (given in Table 3), the chunk test estimates the difference in goodness of fit (by computing the difference in log-likelihood) between the full model (i.e., the combined model from step 3) and the combined model that was built without one studied dimension (whose explanatory power we are computing). The chunk test reports a Chi-square value ($\Delta \text{LR}\chi^2$) (which is the difference in log-likelihood compared to the Chi-squared distribution) and a p-value. The Chi-squared value quantifies the unique explanatory power that was lost due to the removal of the given dimension (in relation to the other dimensions) and a lower p-value ($<= 0.05$) signifies the dimension's significance. We

[13] https://www.rdocumentation.org/packages/Hmisc/versions/4.2-0/topics/redun
[14] https://www.rdocumentation.org/packages/car/versions/3.0-3/topics/Anova
[15] https://www.rdocumentation.org/packages/car/versions/3.0-3

used the `lrtest` function[16] from the `lmtest` package[17] in `R` to conduct the chunk test. Table 4 shows the explanatory power of each dimension (ΔLRχ^2).

5a. **Backward feature selection.** We do backward feature selection to ensure the parsimony of the constructed model, as suggested by Harrell et al. [39]. For instance, if a model contains a large number of independent features, the model becomes too complex to draw explanations. Hence, Harrell et al. [39] suggests using backward feature selection when the goal of the model is to interpret it. We used the `fastbw` function[18] from the `rms` package in `R` to perform a backward elimination of features. The `fastbw` function takes the model that was constructed on all the features (32) and eliminates the features that do not significantly contribute to reducing the AIC of the model. We removed 14 of the 32 features (44%) using the `fastbw` function. In result, we obtained a new combined model with 18 features.

5b. **Build the final model.** With the reduced feature set from step 5a, we reconstructed the final combined model. Similar to step 3, we adjusted the regression coefficients with the bootstrap estimate, as outlined by Harrell et al. [39].

5c. **Performance evaluation.** To demonstrate the quality of the constructed model from 5b, we calculated the AUC of the model using 100 out-of sample bootstrap iterations to evaluate the performance of the model.

5d. **Nomogram analysis.** We used the final combined model from step 5b to create and analyze a nomogram using the `nomogram` function[19] from the `rms` package in `R`, which provides a way to measure the explanatory power of each feature in distinguishing popular from unpopular mods. A nomogram provides a graphical visualization of the parsimonious logistic regression model that we built in step 5b. Although the Wald χ^2 can provide insight into the explanatory power of each feature in the combined model, the nomogram provides us with an exact interpretation on how the variation in each feature affects the outcome probability. For instance, while the Wald χ^2 may indicate that the number of words in the long description of a mod is important, the Wald χ^2 does not provide insights on how the exact number of words in the long description contribute to the explanatory power in distinguishing popular from unpopular mods. Furthermore, the Wald χ^2 does not show if a certain feature has a positive or negative role in distinguishing popular from unpopular mods, whereas the nomogram does. For instance, if for a given mod, the feature "*latest_num_bukkit_versions*" is 0, then it has a positive role in distinguishing popular from unpopular mods. Several prior studies [22, 82] showed that nomograms are one of the most accurate discriminatory tools in interpreting a logistic regression model. Hence, we constructed a nomogram to observe the exact role of features in classifying if a given mod is either popular or unpopular. Another key difference between the Wald χ^2 and nomogram is that the nomogram can show the contribution of each feature towards the outcome probability for each of the studied mods,

[16] https://www.rdocumentation.org/packages/lmtest/versions/0.9-37/topics/lrtest

[17] https://www.rdocumentation.org/packages/lmtest/versions/0.9-37

[18] https://www.rdocumentation.org/packages/rms/versions/5.1-3.1/topics/fastbw

[19] https://www.rdocumentation.org/packages/rms/versions/5.1-3.1/topics/nomogram

Table 4: An overview of the statistics of each dimension and its features. The larger the ΔLRχ^2, the larger the role of a studied dimension. Similarly, the larger the Wald χ^2, the larger the explanatory power of a feature in the combined model (The percentages and p-value are rounded to two decimal places). The feature is statistically significant if the p-value $<= 0.05$. Sorted by the Wald χ^2 per studied dimension.

	Wald χ^2 (%)	**P-value**
Mod Category (ΔLRχ^2: 23.51%)		0.00
Fabric	20.54	< 0.01
Armor, tools, and weapons	7.92	< 0.01
Addons	2.22	< 0.01
Food	2.17	< 0.01
World generation	1.64	< 0.01
API and library	1.47	< 0.01
Miscellaneous	1.39	< 0.01
Server utility	1.17	< 0.01
Storage	0.79	0.02
Redstone	0.50	0.06
Adventure and RPG	0.43	0.08
Cosmetic	0.27	0.17
Technology	0.10	0.39
Map and information	0.02	0.73
Magic	0.01	0.74
Twitch integration	0.00	0.97
Mod Documentation (ΔLRχ^2: 13.03%)		0.00
Number of words in the long description	5.18	< 0.01
Number of images	2.06	< 0.01
Mod wiki URL	1.72	< 0.01
Number of words in the short description	0.46	0.07
Environmental Context of the Mod (ΔLRχ^2: 18.63%)		0.00
Latest number of Bukkit versions	15.45	< 0.01
Latest number of required dependencies	9.27	< 0.01
Latest number of optional dependencies	0.81	0.02
Latest number of Minecraft versions	0.49	0.07
Latest number of Java versions	0.06	0.53
Latest number of tool dependencies	0.00	0.85
Latest number of incompatible dependencies	0.00	0.86
Latest number of embedded library dependencies	0.00	0.98
Remuneration for the Mod (ΔLRχ^2: 10.42%)		< 0.01
Paypal URL	6.21	< 0.01
Patreon URL	1.56	< 0.01
Community Contribution for the Mod (ΔLRχ^2: 34.41%)		0.00
Issues URL	11.21	< 0.01
Source code URL	4.86	< 0.01
Total	100.00	

whereas the Wald χ^2 only shows the overall contribution (which is not specific to each mod). Figure 5 shows the results of the nomogram analysis.

5e. **Partial effects analysis.** We used the final combined model from step 5b and the nomogram analysis from step 5d to create partial effects plots, which show how different values in numeric features with respect to another feature held constant at the median for numeric features and at the mode for boolean features, contributes the outcome probability. Hence, the partial effects analysis provides a deeper explanation of how the variation in certain features can contribute to the probability of a mod being popular or unpopular.

In addition, to measure if two distributions are significantly different, we used the Wilcoxon tests. The Wilcoxon signed-rank test is a paired and non-parametric statistical test, whereas the Wilcoxon rank-sum test is an unpaired and non-parametric statistical test, where the null hypothesis indicates that it is equally likely that a randomly selected value from one sample will be less than or greater than a randomly selected value from a second sample [94]. If the p-value of the used Wilcoxon test on the two distributions is less than 0.05, we reject the null hypothesis, and conclude that the two distributions are significantly different. In addition, to calculate the magnitude of the difference we calculate the Cliff's delta d effect size [54], with the following thresholds [77]:

$$\text{Effect size} = \begin{cases} \textit{negligible(N)}, & \text{if } |d| \leq 0.147. \\ \textit{small(S)}, & \text{if } 0.147 < |d| \leq 0.33. \\ \textit{medium(M)}, & \text{if } 0.33 < |d| \leq 0.474. \\ \textit{large(L)}, & \text{if } 0.474 < |d| \leq 1. \end{cases}$$

Findings: Mods that simplify mod development are a popular type of mods. Figure 5 shows that mods that belong to the *"fabric"*, *"addons"*, and *"API and library"* categories tend to be among the most popular mods. We further investigated the mods under each category and observed that all of the 16 collected *"fabric"* mods are popular mods, 73.3% of the studied *"addons"* mods are popular mods, and 71.1% of the studied *"API and library"* category mods are popular mods. Mods of the *"fabric"* category are created using the *"fabric"* mod development toolchain, which offers a mod ecosystem that makes updating a mod simpler and provides modularity of the code [27]. Mods of the *"API and library"* category can be leveraged by others and mod developers to make mod development simpler. In addition, mods of the *"addons"* category, such as the TOP Addons mod, add support to and extend other mods. [20] Finally, the *"miscellaneous"*, *"food"*, *"world generation"*, *"armor tools weapons"*, and *"server utility"* mod categories are more related to unpopular mods.

Over 70% of the studied popular mods include a source code URL and/or issue tracking URL, as shown in Figure 5. We investigated the studied popular mods and observed that 77% of the popular mods have an issue tracking URL, and 71% of the popular mods have a source code URL. In addition, Figure 6d shows that the presence of an issue tracking URL with at least about 145 words in the mod's main description increases the probability of distinguishing popular from unpopular mods.

[20] https://www.curseforge.com/minecraft/mc-mods/top-addons

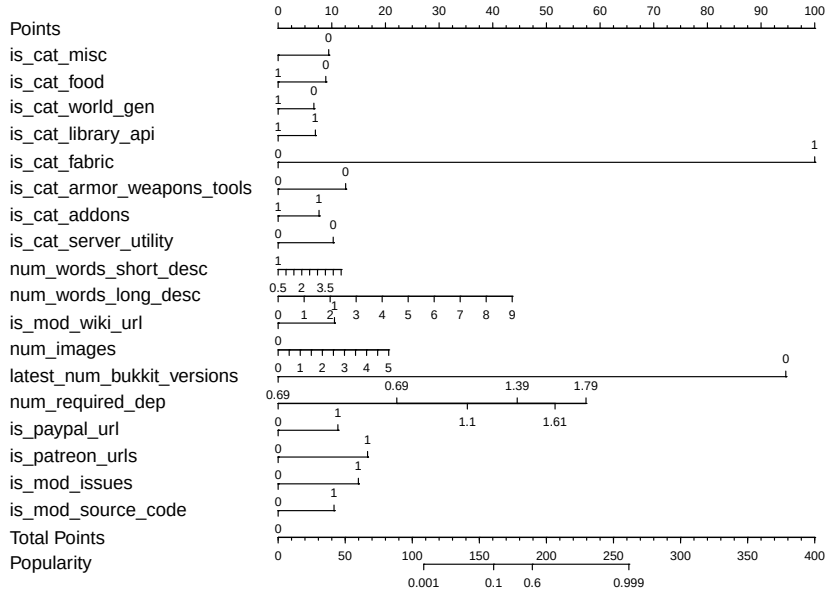
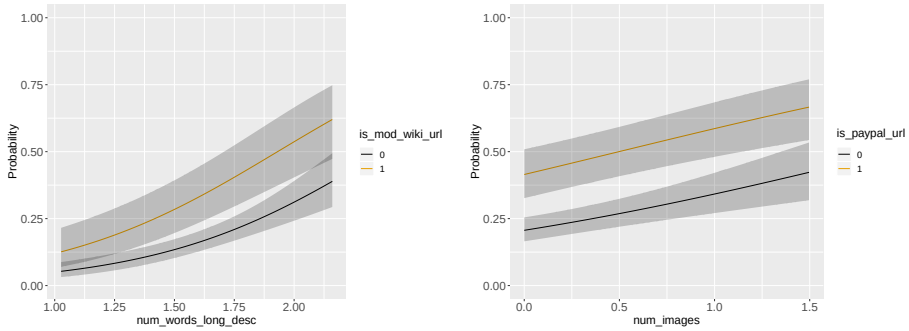


Fig. 5: The nomogram visualizes the role of each feature in distinguishing a mod's popularity. The line against each feature in the figure, varies between the range of values for that given feature. The "*points*" line at the top of the figure, is used to calculate the magnitude of contribution that each feature has and "*Total Points*" at the bottom of the figure gives the total points generated by all the features for a given instance (i.e., for a given mod). For instance, if for a given mod, the feature "*is_cat_fabric*" has a value of 1, then it contributes 100 points. Finally, the line against "*Popularity*" shows the probability of a mod to be classified as a popular mod according to the total number of points (which is computed by summing up all the individual points contributed by each feature). For instance, if all the features for a given mod contribute a total of 260 points, then the probability of that mod to be classified as popular by our explanatory model is 99% and similarly, if the total points given by all the features for a particular mod is less than 110, then that mod will be classified as not popular. Also, the model used to generate this nomogram achieved a median AUC of 0.92 on 100 out-of-sample bootstrap iterations.

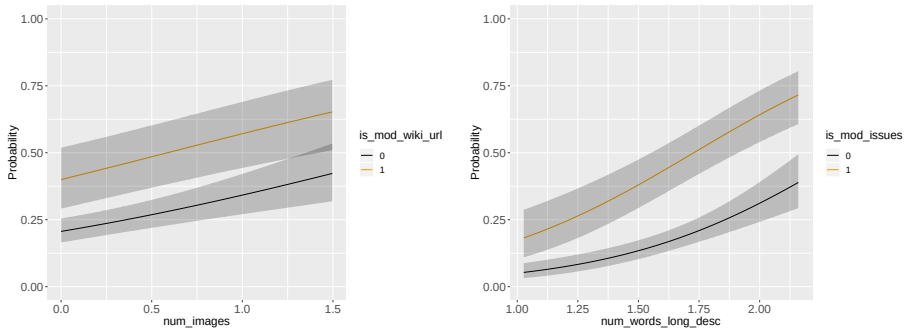
Furthermore, from Table 4, we observe that the community contribution dimension (which captures the presence/absence of source code URL and/or an issue tracking URL) has the highest explanatory power (34.4%) among all the other studied dimensions. Even though other individual features contribute towards characterizing the popularity of a mod, community contribution dimension as a whole is more important.

Popular mods have longer descriptions than unpopular mods. The descriptions of popular mods have a median of 161.5 words, whereas the descriptions of unpopular mods have a median of 75 words. The Wilcoxon rank-sum test confirms that the number of words in the description of popular mods and unpopular mods



(a) Number of words in the long description w.r.t. a mod wiki URL.

(b) Number of images w.r.t. a PayPal URL.



(c) Number of images w.r.t. a mod wiki URL.

(d) Number of words in the long description w.r.t. an issue tracking URL.

Fig. 6: The impact of features on the outcome probability when another feature is held constant (features are held constant at the median for numeric features and at the mode for boolean features). The grey area shows a confidence interval at 95%.

is statistically significantly different, with a medium Cliff's delta effect size. In Figure 6a, we held the mod wiki URL at a constant against the number of words in the description because if a mod developer is willing to provide external documentation, they could be more willing to make an effort into providing a richer description for the mod. Prior work [92] showed that high-rated mobile apps had significantly longer app descriptions, which is consistent with our results.

In addition, Figure 5 shows that popular mods have more images and a wiki URL. Therefore we posit that mod developers who make an effort to provide external documentation are likely to further explain how the mod works visually to users by presenting in-game screenshots, and Figure 6c confirms this observation. Prior work [92] observed that the number of images is one of the top three influential factors in determining that a mobile app will be high-rated, which is consistent with the results of our study of mods. Finally, the number of words in the description, the number of

images, and having a wiki URL are all features that are related to the mod documentation dimension, and all of them have a positive relationship with mod popularity.

Popular mods typically accepted donations and tended to be more active (i.e., they have more releases and comments). Figure 5 and 6b show that popular mods often have a Paypal URL or Patreon URL. Mods with a PayPal URL have a median of 13 mod releases, whereas mods without a PayPal URL have a median of 2 mod releases; mods with a Patreon URL had a median of 21 mod releases, whereas mods without a Patreon URL had a median of 3 mod releases. The Wilcoxon rank-sum test confirms that the differences in the number of mod releases between mods with and without a PayPal URL or Patreon URL are both statistically significant, with a medium Cliff's delta effect size for a PayPal URL and a large Cliff's delta effect size for a Patreon URL. Furthermore, mods with a Patreon URL have a median of 25 comments per mod, while mods without a Patreon URL have a median of 1 comment per mod. The Wilcoxon rank-sum test confirms a statistically significant difference in the number of comments between mods with and without a Patreon URL, with a small Cliff's delta effect size.

In total, we observed that 88 mod developers advertise their Patreon URL on their mods' pages. We manually investigated the motivation of them accepting donations by looking at each of their Patreon profiles. 14% of these mod developers created a Patreon to support their living (e.g., pay bills), 32% of them created a Patreon for fun and did not expect profit, 32% of them created a Patreon to obtain motivation in continuously releasing new content (e.g., faster release of content), and 23% of them either closed or did not finish setting up their Patreon profile.

We further investigated the release frequency of mods (with more than 1 mod release) that are created by the 32% of mod developers who use Patreon for motivation to release new content. However, the Wilcoxon rank-sum test shows no statistically significant difference in the release frequency between mods that are created by mod developers that accept donations for motivation to mod (a median mod release frequency of every 6 days) and mods that are created by other mod developers (a median mod release frequency of 7 days). The Wilcoxon rank-sum test did show a statistically significant difference in the number of mod releases between mods that are created by mod developers that accept donations to mod (a median number of 23 mod releases) and mods that are created by other mod developers (a median number of 11 mod releases), with a medium Cliff's delta effect size. Hence, mod developers who accept donations as a motivation to create mods do produce a larger number of mods than other mod developers (though not necessarily more popular mods). However, their release frequency is similar to the mod developers who do not accept donations as a motivation.

Interestingly, *LexManos*[21] received the most donations at $2,157 per month. LexManos is the creator and primary developer of the popular Minecraft Forge API [60], which is a mod loader API that is required to run most Minecraft mods. However, other mod developers who have a valid Patreon URL only generate a median of $4 per month.

[21] https://www.patreon.com/lexmanos

18 of the 33 (54.5%) studied features have a role in distinguishing popular mods from unpopular ones. Popular mods tend to promote community contributions with a source code URL and an issue tracking URL, and have a richer mod description.

6 Threats to Validity

This section outlines the threats to the validity of our findings.

6.1 Internal Validity

A threat to the internal validity of our study is that we only studied the top and bottom 20% of the mods (based on their number of downloads). However, the top and bottom 20% of the mods ensures that there is a clear distinction between popular and unpopular mods, as mods having close to the median number of total downloads can belong to either one. Such approach is also used in prior study [92].

Another threat to the internal validity of our study is that we only focused on the mods that were created between 2014 and 2016. However, such restriction is necessary to reduce the bias introduced by the extreme short or long lifetime of a mod.

An additional internal threat to validity is that we do not cover all the possible features that are related to mods. However, we conduct a first study to understand the characteristics of popular and unpopular mods specific to a particular game (Minecraft) and we encourage future work to explore additional features and dimensions. For example, Minecraft has been used as a sandbox for a plethora of activities, for example, in the education sector. Therefore, the educational value of a mod might potentially be an important confounder in determining the popularity of a mod in addition to the features that we observe in our study. We suggest that future studies investigate how the other latent functional and educational aspects of Minecraft modding affect its popularity using statistical procedures that are similar to the ones that are outlined in our study.

Finally, it is important to realize that mod developers of the CurseForge mod distribution platform could at anytime change the name of their mod, remove mod developers or delete the mod. As a result, some older mods or mod developers may not exist at the time of our data collection. Future studies should investigate the life cycle of mods and mod developers on the CurseForge mod distribution platform.

6.2 External Validity

A threat to the external validity of our study is that we only studied mods from the CurseForge mod distribution platform. However, the CurseForge mod distribution platform has the largest number of mods out of other mod distribution platforms, as shown in Section 2. Furthermore, we clearly document the data collection and the

statistical approach that we use to arrive at the characteristics of popular game mods in the CurseForge platform. Therefore, our approach could be replicated by other future studies that seek to investigate the characteristics of popular and unpopular mods across different mod distribution platforms (such as the Nexus mods platform). Another threat to the external validity of our study is that we only studied mods for the Minecraft game. Although the Minecraft game is one of the best selling games in 2019, and hosts one of most active and largest modding communities, our results may or may not generalize across mods developed for a different game. Therefore, future studies should use our outlined approach compare our results with mods of different games.

7 Conclusion

An active modding community not only helps game developers meet the growing and changing needs of their gamer base, but also leads to a better overall gaming experience. In this paper, we studied the characteristics of popular mods with a large number of downloads by analyzing 2,228 Minecraft mods from the CurseForge mod distribution platform, along 5 dimensions of characteristics for a mod: mod category, mod documentation, environmental context of the mod, remuneration for the mod, and community contribution for the mod. We firstly verified that the studied dimensions have significant explanatory power in distinguishing popular from unpopular mods. Then, we investigated the contribution of each of the 33 features across these 5 dimensions of mod characteristics on the popularity of a mod. The most important findings of our paper are:

1. The community contribution for the mod dimension has the strongest explanatory power of the popularity of mods. Popular mods tend to promote community contribution with a source code URL and an issue tracking URL.
2. Simplifying the mod development is positively correlated with mod popularity.
3. Popular mods tend to have a high quality description.

Based on our findings, we suggest future work to further investigate the impact of the features that distinguish popular mods, to eventually come with recommendations that assist mod developers in improving the popularity of their mods.

References

1. Ahmed F, Zia M, Mahmood H, Al Kobaisi S (2017) Open source computer game application: An empirical analysis of quality concerns. Entertainment Computing 21:1–10
2. Ahn S, Kang J, Park S (2017) What makes the difference between popular games and unpopular games? analysis of online game reviews from steam platform using word2vec and bass model. ICIC Express Letters pp 1729–1737

3. Al-Washmi R, Bana J, Knight I, Benson E, Kerr OAA, Blanchfield P, Hopkins G (2014) Design of a math learning game using a Minecraft mod. In: European Conference on Games Based Learning, Academic Conferences International Limited, vol 1, p 10
4. Ampatzoglou A, Stamelos I (2010) Software engineering research for computer games: A systematic review. Information and Software Technology 52(9):888–901
5. Arakji RY, Lang KR (2007) Digital consumer networks and producer-consumer collaboration: Innovation and product development in the video game industry. Journal of Management Information Systems 24(2):195–219
6. Balogh G, Beszédes Á (2013) CodeMetrpolis–A Minecraft based collaboration tool for developers. In: 2013 First IEEE Working Conference on Software Visualization (VISSOFT), IEEE, pp 1–4
7. Bavota G, Linares-Vasquez M, Bernal-Cardenas CE, Di Penta M, Oliveto R, Poshyvanyk D (2014) The impact of api change-and fault-proneness on the user ratings of android apps. IEEE Transactions on Software Engineering 41(4):384–407
8. Bayliss JD (2012) Teaching game AI through Minecraft mods. In: International Games Innovation Conference, IEEE, pp 1–4
9. Bebbington S (2014) A case study of the use of the game Minecraft and its affinity spaces for information literacy development in teen gamers. PhD thesis, Université d'Ottawa/University of Ottawa
10. Bécares JH, Valero LC, Martín PPG (2017) An approach to automated videogame beta testing. Entertainment Computing 18:79–92
11. Beggs B (2012) Minecraft, it's a mod, mod, modder's world: Computer game modifications as civic discourse. Reconstruction: Studies in Contemporary Culture 12(2)
12. Blackburn J, Kourtellis N, Skvoretz J, Ripeanu M, Iamnitchi A (2014) Cheating in online games: A social network perspective. ACM Transactions on Internet Technology (TOIT) 13(3):9
13. Blake, Vikki (2019) Minecraft might be the biggest-selling video game of all time now. https://www.eurogamer.net/articles/2019-05-19-minecraft-might-be-the-biggest-selling-video-game-of-all-time-now, (last visited: July 11, 2019)
14. Blincoe K, Sheoran J, Goggins S, Petakovic E, Damian D (2016) Understanding the popular users: Following, affiliation influence and leadership on github. Information and Software Technology 70:30–39
15. Borges H, Valente MT (2018) Whats in a github star? understanding repository starring practices in a social coding platform. Journal of Systems and Software 146:112–129
16. Borges H, Hora A, Valente MT (2016) Predicting the popularity of github repositories. In: Proceedings of the The 12th International Conference on Predictive Models and Data Analytics in Software Engineering, pp 1–10
17. Borges H, Hora A, Valente MT (2016) Understanding the factors that impact the popularity of github repositories. In: 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), IEEE, pp 334–344

18. Brand J, Kinash S (2013) Crafting minds in Minecraft. Education Technology Solutions 55:56–58
19. Canossa A, Martinez JB, Togelius J (2013) Give me a reason to dig Minecraft and psychology of motivation. In: 2013 IEEE Conference on Computational Inteligence in Games (CIG), IEEE, pp 1–8
20. Cheung GK, Zimmermann T, Nagappan N (2014) The first hour experience: how the initial play can engage (or lose) new players. In: First ACM SIGCHI Annual Symposium on Computer-Human Interaction in Play, ACM, pp 57–66
21. Chia PH, Yamamoto Y, Asokan N (2012) Is this app safe? a large scale study on application permissions and risk signals. In: Proceedings of the 21st international conference on World Wide Web, pp 311–320
22. Chun FKH, Karakiewicz PI, Briganti A, Walz J, Kattan MW, Huland H, Graefen M (2007) A critical appraisal of logistic regression-based nomograms, artificial neural networks, classification and regression-tree models, look-up tables and risk-group stratification models for prostate cancer. BJU international 99(4):794–800
23. CurseForge (2006) CurseForge. https://minecraft.curseforge.com/, (last visited: March 12, 2019)
24. Dey T, Massengill JL, Mockus A (2016) Analysis of popularity of game mods: A case study. In: Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts, ACM, pp 133–139
25. Duncan SC (2011) Minecraft, beyond construction and survival. Well Played: a journal on video games, value and meaning 1(1):1–22
26. Ekaputra G, Lim C, Eng KI (2013) Minecraft: A game as an education and scientific learning tool. Information Systems International Conference (ISICO)
27. Fabric development team (2018) Fabric Announcement. http://fabricmc.net/2018/12/10/announcement.html, (last visited: August 30, 2019)
28. Finley, Klint (2014) New Minecraft Mod Teaches you Code as you Play. https://www.wired.com/2014/08/learntomod/, (last visited: July 11, 2019)
29. Fox J, Monette G (2002) An R and S-Plus companion to applied regression. Sage
30. Geere D, Copeland W (2019) The best Minecraft mods. https://www.pcgamer.com/best-minecraft-mods/, (last visited: July 11, 2019)
31. Graham TN, Roberts W (2006) Toward quality-driven development of 3D computer games. In: International Workshop on Design, Specification, and Verification of Interactive Systems, Springer, pp 248–261
32. Guana V, Stroulia E, Nguyen V (2015) Building a game engine: A tale of modern model-driven engineering. In: IEEE/ACM 4th International Workshop on Games and Software Engineering, IEEE, pp 15–21
33. Guerrouj L, Azad S, Rigby PC (2015) The influence of app churn on app success and stackoverflow discussions. In: 2015 IEEE 22nd International Conference on Software Analysis, Evolution, and Reengineering (SANER), IEEE, pp 321–330
34. Hackman E, Björkqvist U (2014) Modders of Skyrim: Motivations and Modifications: A qualitative study of what motivations and modifications the modders of Elder Scrolls: Skyrim exhibit. Master's thesis, Södertörn University
35. Hanghøj T, Hautopp H, Jessen C, Denning RC (2014) Redesigning and reframing educational scenarios for Minecraft within mother tongue education. In: Eu-

ropean Conference on Games Based Learning, Academic Conferences International Limited, vol 1, p 182
36. Harman M, Jia Y, Zhang Y (2012) App store mining and analysis: Msr for app stores. In: 2012 9th IEEE working conference on mining software repositories (MSR), IEEE, pp 108–111
37. Harrell Jr FE (2001) Lecture Notes in Biostatistical Modeling. URL: http://biostat.mc.vanderbilt.edu/wiki/pub/Main/BioMod/notes.pdf. Last visited: September 2, 2019
38. Harrell Jr FE, Slaughter JC (2001) Introduction to Biostatistics for Biomedical Research. URL: http://hbiostat.org/doc/bbr.pdf. Last visited: September 2, 2019
39. Harrell Jr FE, Lee KL, Califf RM, Pryor DB, Rosati RA (1984) Regression modelling strategies for improved prognostic prediction. Statistics in medicine 3(2):143–152
40. Jeppesen LB (2004) Profiting from innovative user communities: How firms organize the production of user modifications in the computer games industry. Tech. rep., Department of Industrial Economics and Strategy, Copenhagen Business School
41. Kalliamvakou E, Gousios G, Blincoe K, Singer L, German DM, Damian D (2014) The promises and perils of mining github. In: Proceedings of the 11th working conference on mining software repositories, pp 92–101
42. Köhler B, Haladjian J, Simeonova B, Ismailović D (2012) Feedback in low vs. high fidelity visuals for game prototypes. In: Second International Workshop on Games and Software Engineering: Realizing User Engagement with Game Engineering Techniques, IEEE, pp 42–47
43. Lane HC, Yi S, Guerrero B, Comins NF (2017) Minecraft as a sandbox for stem interest development: Preliminary results. In: 25th International Conference on Computers in Education Proceedings
44. Leavitt A (2013) The source of open-source culture: Participation in the production of an open media artifact, minecraft. AoIR Selected Papers of Internet Research 3
45. Lee D, Lin D, Bezemer CP, Hassan AE (2018) Building the perfect game - an empirical study of game modifications. Empirical Software Engineering Under review:1–23
46. Lenig S, Caporusso N (2018) Minecrafting virtual education. In: International Conference on Applied Human Factors and Ergonomics, Springer, pp 275–282
47. Lewis C, Whitehead J (2011) The whats and the whys of games and software engineering. In: 1st International Workshop on Games and Software Engineering, ACM, pp 1–4
48. Lewis C, Whitehead J, Wardrip-Fruin N (2010) What went wrong: a taxonomy of video game bugs. In: Fifth International Conference on the Foundations of Digital Games, ACM, pp 108–115
49. Lin D, Bezemer CP, Hassan AE (2017) Studying the urgent updates of popular games on the Steam platform. Empirical Software Engineering 22(4):2095–2126
50. Lin D, Bezemer CP, Hassan AE (2018) An empirical study of early access games on the Steam platform. Empirical Software Engineering 23(2):771–799

51. Lin D, Bezemer CP, Hassan AE (2019) Identifying gameplay videos that exhibit bugs in computer games. Empirical Software Engineering 24(115):1573–7616
52. Lin D, Bezemer CP, Zou Y, Hassan AE (2019) An empirical study of game reviews on the Steam platform. Empirical Software Engineering 24(1):170–207
53. Linares-Vásquez M, Bavota G, Bernal-Cárdenas C, Di Penta M, Oliveto R, Poshyvanyk D (2013) Api change and fault proneness: a threat to the success of android apps. In: Proceedings of the 2013 9th joint meeting on foundations of software engineering, pp 477–487
54. Long JD, Feng D, Cliff N (2003) Ordinal analysis of behavioral data. Handbook of psychology pp 635–661
55. McIntosh S, Kamei Y, Adams B, Hassan AE (2016) An empirical study of the impact of modern code review practices on software quality. Empirical Software Engineering 21(5):2146–2189
56. Midi H, Sarkar SK, Rana S (2010) Collinearity diagnostics of binary logistic regression model. Journal of Interdisciplinary Mathematics 13(3):253–267
57. Mills, Aaron (2015) A Brief History of Minecraft Modding. https://hub.packtpub.com/brief-history-minecraft-modding/, (last visited: July 11, 2019)
58. Mojang (2019) What is Minecraft? https://www.minecraft.net/en-us/what-is-minecraft/, (last visited: July 11, 2019)
59. Molnar C (2018) Interpretable machine learning. A Guide for Making Black Box Models Explainable 7, URL https://christophm.github.io/interpretable-ml-book/
60. Morrison A (2014) How To Make Minecraft A Survival Game. https://www.rockpapershotgun.com/2014/10/24/minecraft-survival-mods/, (last visited: March 12, 2019)
61. Müller S, Kapadia M, Frey S, Klinger S, Mann RP, Solenthaler B, Sumner RW, Gross M (2015) Statistical analysis of player behavior in Minecraft. In: Proceedings of the 10th International Conference on the Foundations of Digital Games, Society for the Advancement of the Science of Digital Games
62. Murphy-Hill E, Zimmermann T, Nagappan N (2014) Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? In: 36th International Conference on Software Engineering, ACM, New York, NY, USA, pp 1–11
63. Nagappan M, Shihab E (2016) Future trends in software engineering research for mobile apps. In: 2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER), IEEE, vol 5, pp 21–32
64. Nebel S, Schneider S, Rey GD (2016) Mining learning and crafting scientific experiments: A literature review on the use of minecraft in education and research. Journal of Educational Technology & Society 19(2):355–366
65. Nguyen J (2016) Minecraft and the building blocks of creative individuality. Configurations 24(4):471–500
66. Nieborg DB, Van der Graaf S (2008) The mod industries? the industrial logic of non-market game production. European Journal of Cultural Studies 11(2):177–195

67. O'Brien, Chris (2013) How Minecraft became one of the biggest video games in history. https://www.latimes.com/business/la-xpm-2013-sep-03-la-fi-tn-how-minecraft-video-games-20130822-story.html, (last visited: July 11, 2019)
68. Pascarella L, Palomba F, Di Penta M, Bacchelli A (2018) How is video game development different from software development in open source? In: 2018 IEEE/ACM 15th International Conference on Mining Software Repositories (MSR), pp 392–402
69. Petrillo F, Pimenta M, Trindade F, Dietrich C (2008) Houston, we have a problem... a survey of actual problems in computer games development. In: Proceedings of the 2008 ACM symposium on Applied computing, pp 707–711
70. Petrillo F, Pimenta M, Trindade F, Dietrich C (2009) What went wrong? a survey of problems in game development. Computers in Entertainment (CIE) 7(1):1–22
71. Petrov A (2014) Using Minecraft in education: A qualitative study on benefits and challenges of Game-Based Education. Unpublished masters thesis, University of Toronto, Ontario, Canada URL https://tspace.library.utoronto.ca/bitstream/1807/67048/1/Petrov_Anton_201406_MT_MTRP.pdf
72. Phillips T (2018) The human cost of Red Dead Redemption 2. https://www.eurogamer.net/articles/2018-10-25-the-human-cost-of-red-dead-redemption-2, (last visited: March 12, 2019)
73. Politowski C, Fontoura L, Petrillo F, Guéhéneuc YG (2016) Are the old days gone?: A survey on actual software engineering processes in video game industry. In: 5th International Workshop on Games and Software Engineering, ACM, pp 22–28
74. Poretski L, Arazy O (2017) Placing value on community co-creations: A study of a video game 'modding' community. In: ACM Conference on Computer Supported Cooperative Work and Social Computing, ACM, pp 480–491
75. Postigo H (2007) Of mods and modders: Chasing down the value of fan-based digital game modifications. Games and Culture 2(4):300–313
76. Quiring T (2015) From voxel vistas: Place-making in minecraft. Journal for virtual worlds research 8(1)
77. Romano J, Kromrey JD, Coraggio J, Skowronek J, Devine L (2006) Exploring methods for evaluating group differences on the nsse and other surveys: Are the t-test and cohensd indices the most appropriate choices. In: Annual Meeting of the Southern Association for Institutional Research, Citeseer
78. Ruiz A, Villa N (2008) Storms prediction: Logistic regression vs random forest for unbalanced data. arXiv preprint arXiv:08040650
79. Saito D, Takebayashi A, Yamaura T (2014) Minecraft-based preparatory training for software development project. In: 2014 IEEE International Professional Communication Conference (IPCC), IEEE, pp 1–9
80. Scacchi W, Cooper KM (2015) Research challenges at the intersection of computer games and software engineering. In: Conference on Foundations of Digital Games
81. Scott R (2007) Nexus Mods. https://www.nexusmods.com/, (last visited: August 30, 2019)

82. Shariat SF, Karakiewicz PI, Godoy G, Lerner SP (2009) Use of nomograms for predictions of outcome in patients with advanced bladder cancer. Therapeutic advances in urology 1(1):13–26
83. Short D (2012) Teaching scientific concepts using a virtual world–Minecraft. Teaching Science-the Journal of the Australian Science Teachers Association 58(3):55
84. Shumovsky Y (2018) How much does it cost to make a video game? https://vironit.com/how-much-does-it-cost-to-make-a-video-game/, (last visited: March 12, 2019)
85. Sifa R, Bauckhage C, Drachen A (2014) The playtime principle: Large-scale cross-games interest modeling. In: IEEE Conference on Computational Intelligence and Games, IEEE, pp 1–8
86. Siko J, Barbour M, Toker S (2011) Beyond Jeopardy and lectures: Using Microsoft PowerPoint as a game design tool to teach science. Journal of Computers in Mathematics and Science Teaching 30(3):303–320
87. Stone BG, Mills KA, Saggers B (2019) Online multiplayer games for the social interactions of children with autism spectrum disorder: a resource for inclusive education. International Journal of Inclusive Education 23(2):209–228
88. Taba SES, Keivanloo I, Zou Y, Ng J, Ng T (2014) An exploratory study on the relation between user interface complexity and the perceived quality. In: International Conference on Web Engineering, Springer, pp 370–379
89. Tantithamthavorn C, McIntosh S, Hassan AE, Matsumoto K (2016) An empirical comparison of model validation techniques for defect prediction models. IEEE Transactions on Software Engineering 43(1):1–18
90. Targett S, Verlysdonk V, Hamilton HJ, Hepting D (2012) A study of user interface modifications in World of Warcraft. Game Studies 12(2)
91. Thongtanunam P, Hassan AE (2018) Review dynamics and its impact on software quality. IEEE Transactions on Software Engineering pp 1–13
92. Tian Y, Nagappan M, Lo D, Hassan AE (2015) What are the characteristics of high-rated apps? A case study on free Android applications. In: Software Maintenance and Evolution (ICSME), IEEE International Conference on, IEEE, pp 301–310
93. Washburn Jr M, Sathiyanarayanan P, Nagappan M, Zimmermann T, Bird C (2016) What went right and what went wrong: an analysis of 155 postmortems from game development. In: 38th International Conference on Software Engineering Companion, ACM, pp 280–289
94. Wilcoxon F (1945) Individual comparisons by ranking methods. Biometrics bulletin 1(6):80–83
95. Wu HA (2016) Video game prosumers: Case study of a minecraft affinity space. Visual Arts Research 42(1):22–37
96. Zhu J, Zhou M, Mockus A (2014) Patterns of folder use and project popularity: A case study of github repositories. In: Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement, pp 1–4
97. Zorn C, Wingrave CA, Charbonneau E, LaViola Jr JJ (2013) Exploring Minecraft as a conduit for increasing interest in programming. In: FDG, International Con-

ference on the Foundations of Digital Games, pp 352–359