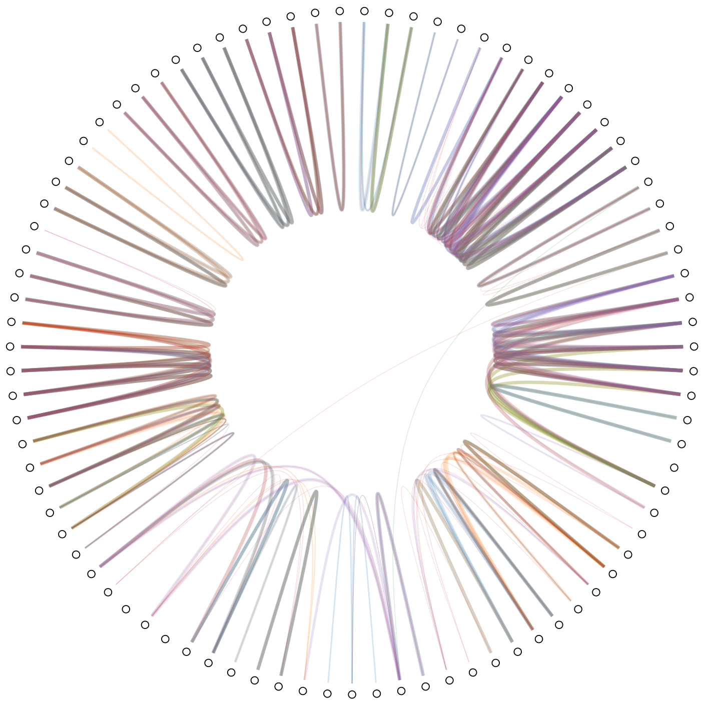
Quantifying and Predicting Collaboration in Shared Virtual Worlds



Stephan Müller

Master Thesis
June 2015

Prof. Dr. Markus Gross

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

computer graphics laboratory

Abstract

We created an open-source data collection and analysis framework for recording and monitoring gameplay in Minecraft, interactive information visualization tools and an analysis framework for players, administrators, and researchers to explore graphs, maps and timelines. We collected a large dataset of recorded gameplay to be used for social studies on shared virtual worlds and introduce an indicator to measure collaboration, which we use to suggest actionable ways to increase collaboration on a Minecraft server. The framework can be used for studying not only "personality" differences between individual players, but "cultural" differences that ultimately emerge between servers.

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

computer graphics laboratory

Master Thesis

Promoting Collaboration in social sandbox games

Figure 1: Two players in Minecraft ready to fight

Introduction

Player collaboration is in important part of playing sandbox games like Minecraft on online servers. It is assumed that collaboration plays a big role in player satisfaction and retaining a servers user base. And yet, apart from a global in-game chat, Minecraft does not not provide any help for players to find collaborators or shared tasks.

Task Description

The goal of this project to understand and increase player collaboration on Minecraft servers. The first step will be to analyze different kinds of collaboration. Existing user data recordings can be used to detect as well as predict player collaboration. Additional user data might have to be collected. The second part will involve affecting collaboration. Polices and tools have to be developed, deployed on servers and their effect on players analyzed. A possible tool might be an in-game compass leading players to certain places.

Skills

- Programming in Java, Python, PHP, R
- Experience in data mining, data visualization.
- (Preferred) Experienced Minecraft player

Remarks

- This project can start as soon as possible
- A written report and an oral presentation conclude the thesis
- This thesis will be supervised by Prof. Markus Gross (ETH) and Dr. Seth Frey (DRZ).

Contact

For further information, please contact Seth Frey (seth.frey@disneyresearch.com), Mubbasir Kapadia (mubbasir.kapadia@disneyresearch.com), Barbara Solenthaler (solenthaler@inf.ethz.ch), and Severin Klingler (severin.klingler@inf.ethz.ch)

Acknowledgments

The authors wish to thank Simon Marti, the Castronova family and Jonathan Yarbor of node-craft.com for their valuable input. We also wish to thank the code reviewers at bukkit.org and Youtube user CraftingBenchMC for creating an introduction video for DiviningRod.

The main author wishes to thank his supervisors Seth Frey, Mubbasir Kapadia, Severin Klingler, Richard Mann and Barbara Solenthaler, for going to great lengths to make this project possible and contributing great ideas and critical input along the way.

Contents

List of Figures

List of Tables

1

Introduction

1.1. Motivation

Collaboration is an indispensable aspect of shared virtual worlds, where players inhabit the same virtual environment, communicate with each other either, and build together. Each player's actions can have longstanding ramifications for the evolution of the world and it's community's experiences.

We define collaboration as working with others on a task or to achieve shared goals, and thereby include almost any interaction players have with each other, either directly or by using or experiencing player-generated changes to the world. Surprisingly, there are no standard measures and predictors for collaboration in shared virtual worlds. Our goal is to fill this gap.

In the process of doing so, we address another problem in game analytics research, namely that it tends to serve only game studios, with a focus on monetization, retention, and other questions of strictly commercial interest. But analytics can also serve players by helping them monitor and maintain healthy communities. Consistent with our general approach of improving collaboration by empowering players, our system democratizes game analytics with an analytic framework that can serve the needs of both researchers and players.

1.2. Symbiosis with Server Administrators

Administrators of shared virtual worlds are interested in building strong communities of recurring players. Collaboration is not only an indicator for strong communities, it is what players expect to experience when joining a shared virtual world. A way to quantify collaboration allows administrators to measure their success. Better understanding and being able to predict collaboration helps them reach their goal.

On the other hand, we need large datasets of player behavior in order to make generalizable observations about player and their interactions. We recruited many administrators to contribute data to our research by providing them with research-based tools for monitoring social aspects of their servers.

1.3. Contributions

This thesis makes the following main contributions:

- An open-source data collection and analysis framework for recording and monitoring gameplay in Minecraft
- Interactive information visualization tools and an analysis framework for players, administrators, and researchers to explore graphs, maps and timelines
- A large dataset of recorded gameplay to be used for social studies on shared virtual worlds
- An indicator to measure collaboration, which we use to suggest actionable ways to increase collaboration on a Minecraft server
- A framework for studying not only "personality" differences between individual players, but "cultural" differences that ultimately emerge between servers.

1.4. Results

We collected over 3452 hours of gameplay from 43 different servers. Despite our interest in collaboration, the recorded data includes most aspects of a player's gameplay in great detail and can be used to study aspects beyond collaboration.

We found player contact to be a good indicator for a wide range of collaboration types. In addition to detecting collaboration, we were able to appropriately quantify collaboration using contact duration and frequency. We introduce the collaboration index, defined as the average number of other players a player is in contact with over time. We show that this simple measure is highly correlated with several different estimators of collaboration, each of which is either less general or more complicated. The collaboration index provides a convenient way of comparing amounts of collaboration between different players or servers.

Significant correlations in our collected data suggest that players are more likely to collaborate with players they already collaborated with. Players increase their amount of collaboration not by collaborating with the same player more often, but by collaborating longer and with more different players. We found that players who spend a lot of time fighting collaborate more and players who prefer mining collaborate less. We also found collaborative players to be more socially responsible by giving more to shared resources than they take.

Players prefer to team up with players showing similar interests in respect to building, mining, fighting and exploring. Players with a strong preference for fighting or building are especially likely to collaborate with other fighters or builders, respectively. Players doing much fighting but little building tend to team up with their opposites and players with a high specialization on exploring tend to team up with more balanced players. Differences in experience levels have no effect on the likelihood of teaming up.

1.5. Related Work

The scientific study of human collaboration was an innovation that was motivated by new communications technologies — indeed, the first laboratory experiments with groups were not run in psychology or sociology laboratories, but in the electronics and telecommunications laboratories of 1950s MIT, Harvard University, and Carnegie Mellon University [Bav50, HM51, GS55].

Today, another technology is motivating innovation in the quantitative study of collaboration. The online societies of the internet are leveraging the anonymity and low transaction costs of the medium to test new structures and accomplish new kinds of feats. Understanding collaboration in them can make them work more effectively, and it can also teach us about collaboration more generally. This task is fundamentally interdisciplinary, and must necessarily engage computer scientists, psychologists, sociologists, and others.

1.5.1. Collaboration in Shared Virtual Worlds

The online social systems that have attracted the most attention from quantitative collaboration researchers are wikis, massively multiplayer online (“MMO”) games, and other shared virtual worlds. The most popular domain for studies of online collaboration has been the wiki, particularly Wikipedia [Oko09, Wag04], but games like Minecraft offer unique opportunities for advancing such research. We focus specifically on research that uses games to study collaboration. The 2013 volume edited by El-Nasr, Drachen, and Canossa [SENDC13] introduces quantitative and qualitative methods for evaluating game play, mainly toward the end of extracting design insights for developers in game studios. Though most of its analytical focus treat individual players as the focal unit of analysis, two of its papers attend to whole player communities. Ducheneut and Yee [DY13] present a framework for collecting behavioral and survey data about players, their avatars, and their guilds, allowing them to compare the differences in the traits at each scale. Castronova et al. [CRK13] used an econometric approach to analyze how World of Warcraft guilds maximize fun. Elsewhere, [Che09] introduces DKP, or “dragon kill points”, as a currency that emerged among collaborators in World of Warcraft for the fair distribution of indivisible goods. Chung et al. [CHC$^+$14] examine a large dataset of players in the Korean MMO “Aion” to identify and characterize successful player groups.

Research on Minecraft

Recent years have seen a growing interest in MInecraft from academic researchers and game practitioners alike, sparked largely by the game's explosive popularity. Its success can be largely attributed to its focus on creativity and shared experiences. Who plays Minecraft? No professionally-produced surveys have been published, but an informal survey of 5000 Reddit users report a median age of approximately 19, of whom 94% were male [Red]. [1]

Most current research work on Minecraft is qualitative, from researchers interested in education or peer production [Gar14][SC13]. Leavitt's ethnography identifies the social media culture surrounding the game as intrinsic to the experience of it [Lea13]. French et al. [FSN+14] used it as a platform for a multiuser CAD application, while Bukvic et al. explored the game's potential for eliciting live creative performance with physical actors [BCW+14].

Collaboration in the real world

In the process of crafting new forums for collaboration online, today's online societies have made it possible to advance our understanding of human affairs offline [Bai07]. Castronova reports natural experiments and other large-scale studies that speak to real-world macro-scale social and economic phenomena [Cas08, Cas06, CWS+09]. The work of Szell and colleagues on the multiplayer game Pardus has made important contributions to our understanding of the interactions between different kinds of social networks [SLT10, MFS+15].

While most previous work has focused on the macro-scale effects of individual behavior, there is increasing interest in human collaboration. The Virtual Worlds Observatory (http://vwobservatory.com/) consortium has used MMO games and worlds for a variety of analyses of both individual motivations and team-scale behavior. Their work, particularly that by Contractor and colleagues, has been applied extensively to the study of organizational dynamics [HSC13, HYBC13]. Duchenaut and Yee [DYNM07] find in a large World of Warcraft corpus that large-scale collaboration is most likely to persist when groups are organized into specialized subgroups that can continuously integrate new members. Going one step further into the real-world implications of online activity, Keegan et al. [KAW+10] used cooperative activity on the virtual world Second Life to detect criminal cartel activity in the real world.

Despite the sophisticated research to date, virtual worlds have only barely expressed their potential to illuminate the mechanisms driving real-world social phenomena. Minecraft stands out in this respect because of a subtle difference from other popular games. Unlike World of Warcraft, Second Life, and League of Legends, most Minecraft servers are owned by players and not by the game's ownership. This has two important implications. First, there are many more independent instantiations of Minecraft worlds than of any other game world. Second, the data produced by a given server are not owned by a corporation that may be reticent to share it with academic researchers, but by players who may be eager to contribute to science. These differences permit Minecraft to make unique contributions to virtual world research.

[1]The survey reported by French et al. [FSN+14] finds a slightly less unbalanced gender distribution of 85%.

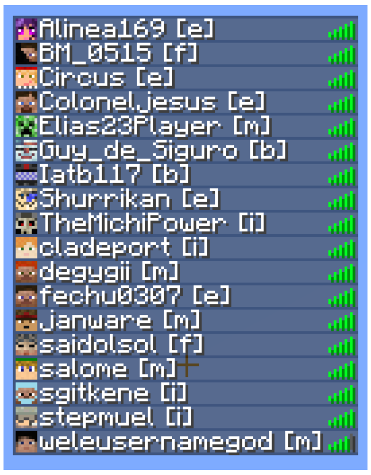
2 HeapCraft

To support our research in studying collaboration in Minecraft, we have developed `HeapCraft`: an open-source suite of tools to record, analyse, and visualize player behavior. Because our tools, and the datasets they have generated, can be used for purposes beyond the study of collaboration, they are of interest to other researchers and server administrators, and are likely to be used into the future. On heapcraft.net, researchers and server admins can find information about our research, instructions on how to use these tools and instructions on how to participate in our data collection. In this section we will introduce parts of the `HeapCraft` project relevant to our research on collaboration.

2.1. Epilog

The Epilog plugin allows server administrators to send player data to our data collection server. The plugin keeps server performance impact to a minimum and sends data over an encrypted connection. Epilog records almost all player-related game events, including player movement, block placement, mining and inventory content. A list of all recorded events can be found in Appendix C.1. The logging of chat messages can be disabled.

The current version of Epilog extends the framework presented in [MKF+ar]. We added additional data collection features and made it easy for amateur server administrators to install and benefit from. A more detailed description can be found in the user manual in Appendix B.1.



```
Alinea169 [e]
BM_0515 [f]
Circus [e]
Coloneljesus [e]
Elias23Player [m]
Guy_de_Siguro [b]
Iatb117 [b]
Shurrikan [e]
TheMichiPower [i]
cladeport [i]
degygii [m]
fechu0307 [e]
janware [m]
saidolsol [f]
salome [m]
sgitkene [i]
stepmuel [i]
weleusernamegod [m]
```

Figure 2.1.: *The list of online players inside Minecraft, annotated by Classify with [b]uild, [m]ine, [f]ight, [e]xplore or [i]dle.*

2.2. Data Processing Backend

In addition to being written to a log file, all data sent to our logging server is processed instantly and added to specialized datasets like aggregated player statistics, player position maps and a history of online players on the server. This preprocessing made it possible to efficiently serve real-time statistics and position heat maps used by other components of `HeapCraft`, like the Epilog visual analytics dashboard. More information about the data processing backend can be found in Appendix A.

2.3. Epilog Dashboard

The Epilog Dashboard is a web-based visual analytics front-end which provides Minecraft server administrators with insight into the collected data. Available datasets include heat maps of player positions and player properties like number of placed blocks, duration of active game-play or time spent near other players. The front- and back- ends work efficiently enough together to permit real-time monitoring of all server activity.

We created the Epilog Dashboard to increase adoption of Epilog by providing additional value to server administrators. The realtime player position heat maps turned out to be a particularly popular feature.

2.4. Classify

The Classify plugin annotates the in-game list of online players by classifying their current behavior as either building, mining, fighting, exploring or idle. The result is shown in Figure 2.1. The classifier is based on the work of our previous paper [MKF$^+$ar] and uses data collected by

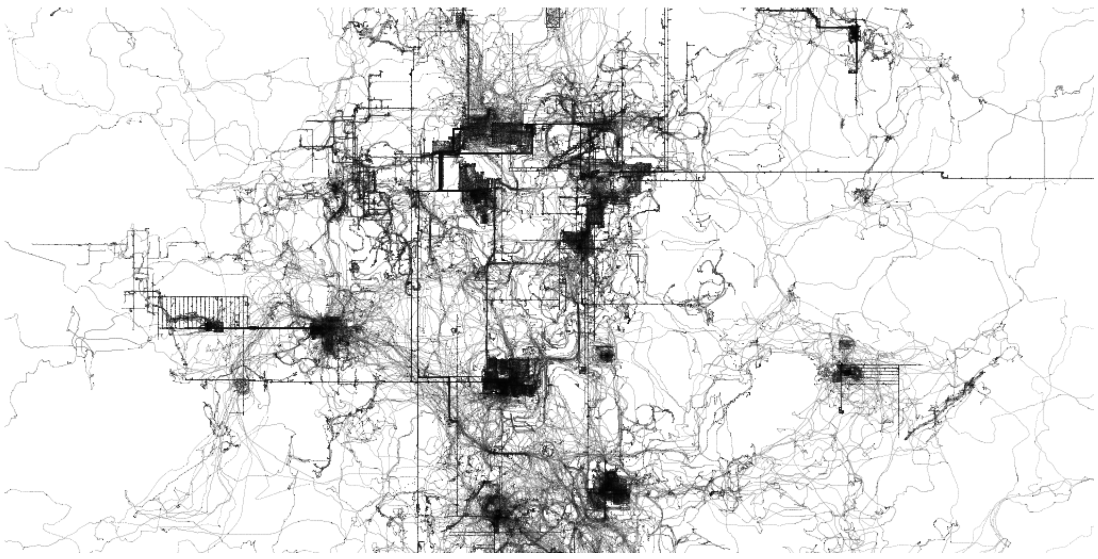


Figure 2.2.: *A heat-map of accumulated player positions over time, produced by the Map Miner tool.*

the Epilog plugin.

We created Classify to increase adoption of Epilog by providing additional value to players. Specifically, Classify was suggested by an experienced Minecraft server administrator as a simple and immediately useful application of the results of our prior work.

2.5. Graph Miner

We developed a web-based tool for exploration of graph data. The tool can be used to compare multiple graphs using the same vertices. Edges and vertices can be sorted and filtered by previously assigned properties and graph metrics using JavaScript expressions. A more detailed description can be found in Appendix B.2.

We used Graph Miner to explore player relationships and to create many of the visualizations herein.

2.6. Map Miner

We have developed a web-based tool for the exploration of spatial data, timelines and relations between them. Data can be visualized on a timeline which allows selection of time periods. A map then shows spatial data for the selected time. The tool provides a better understanding of what is happening on individual servers by looking at player position maps. Both the map and the timeline can be zoomed and dragged to reveal additional details.

3

Data Exploration

3.1. Active Gameplay

Our dataset shows many instances of players remaining inactive for periods of up to several hours while continue to be logged in. This behavior can be caused by players being away from their keyboard or, within Minecraft, by waiting for daytime, for their health to regenerate or for plants to grow. We exclude those periods of inactivity by using active gameplay instead of online state when determine a player's presence.

We defined players as active if they triggered a game event within the last 20 seconds. When measuring a player's active time we start at the first event but exclude the 20 seconds after the last event. This keeps time measurements more accurate in terms of active gameplay. For other matters (like player proximity) we consider a player active from the first event until 20 seconds after the last event. This is not only easier to compute, but also better matches the perception of other players. An avatar can stay inactive for a couple of seconds until it is assumed the player has left the keyboard.

3.2. Data Acquisition

Using the Epilog plugin, we collected 3452 hours of active gameplay by 4763 different players on 43 servers over a period of 65 days between March 13, 2015 and May 17, 2015.

Figure 3.1 shows the distribution of servers with low amounts of recorded gameplay. 18 servers contributed only a few minutes of active gameplay. We assume servers with less than 5 minutes of gameplay are mostly the result of server admins evaluating the Epilog plugin and deciding

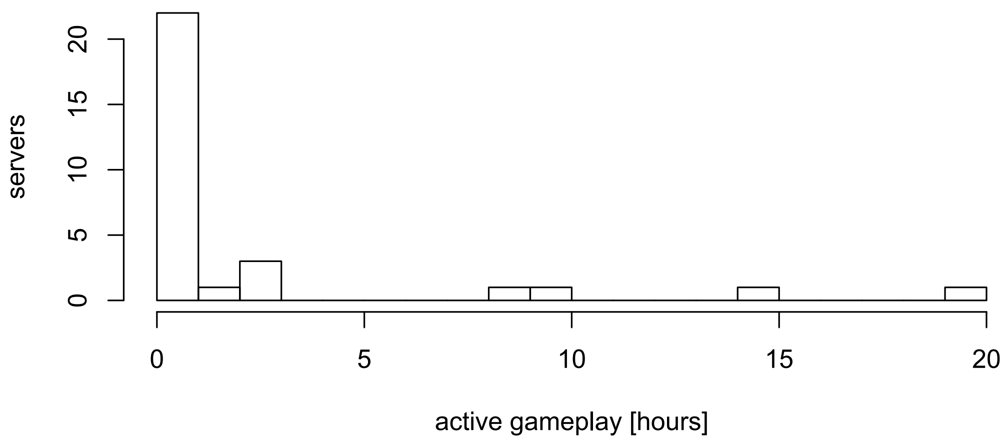


Figure 3.1.: *Histogram of recorded gameplay for servers under 20 hours.*

not to keep it installed. 30 servers have less than 5 hours of recorded gameplay. Of the two servers with just below 10 hours, one had only 3 players and the other shared players with a third much more popular server. Since the latter server stopped sending data before the more popular server started sending data, we assume it to be an evaluation server to test plugins before deploying them.

Considering these data quality factors, we excluded all servers with less than 10 hours of recorded active gameplay from our dataset, leaving us with 15 servers, at the cost of only about 30 player-hours of data.

One server within our restricted dataset of fifteen was an outlier with 4133 unique players. Further inspection showed that there was a spike of thousands of logins over a period of only a few minutes, and that only 266 players were active for longer than one second. We assume this to be caused by either a malfunction of one of the server 's plugins or a denial of service attack.

Figure 3.2 shows the distribution of players with short activity times. Among the 828 players of the 15 servers with more than 1 second of gameplay, 70% are active for less than one hour. The recorded gameplay of those players is likely to not represent their average behavior adequately. The remaining 252 players contribute 3342 hours of gameplay which is 98% of the 3421 hours in total. Players with less than 1 second of gameplay sum up to a total of only one hour while players below 1 hour sum up to 79 hours. Consequently, players with less than one hour of gameplay are excluded from the analyses below unless stated otherwise.

Table 3.1 shows the number of players and the combined amount of active gameplay for each server. It also notes which of our plugins are used. Figure 3.3 shows the number of active players during the data collection period. The server enumeration and the assigned colors are consistent throughout the thesis.

The number of active players on the timeline is capped at 20. The alleged denial of service

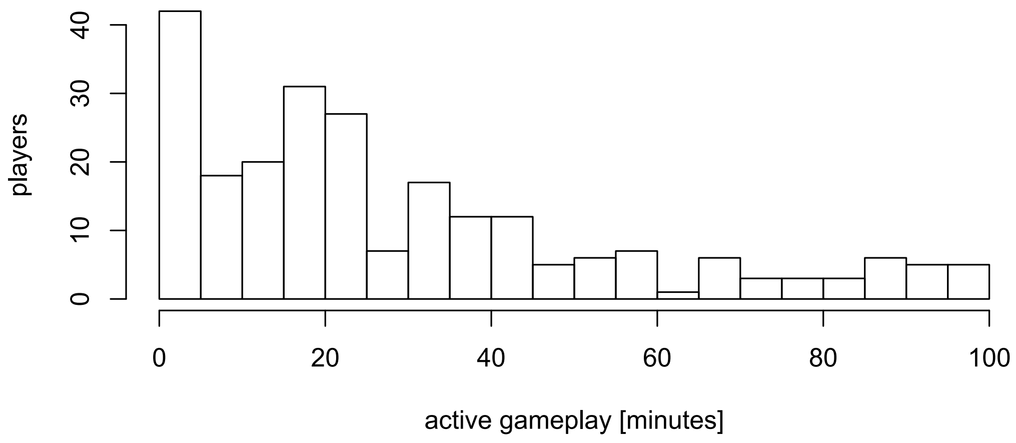


Figure 3.2.: *Histogram of recorded gameplay for players under 100 minutes but over 1 second.*

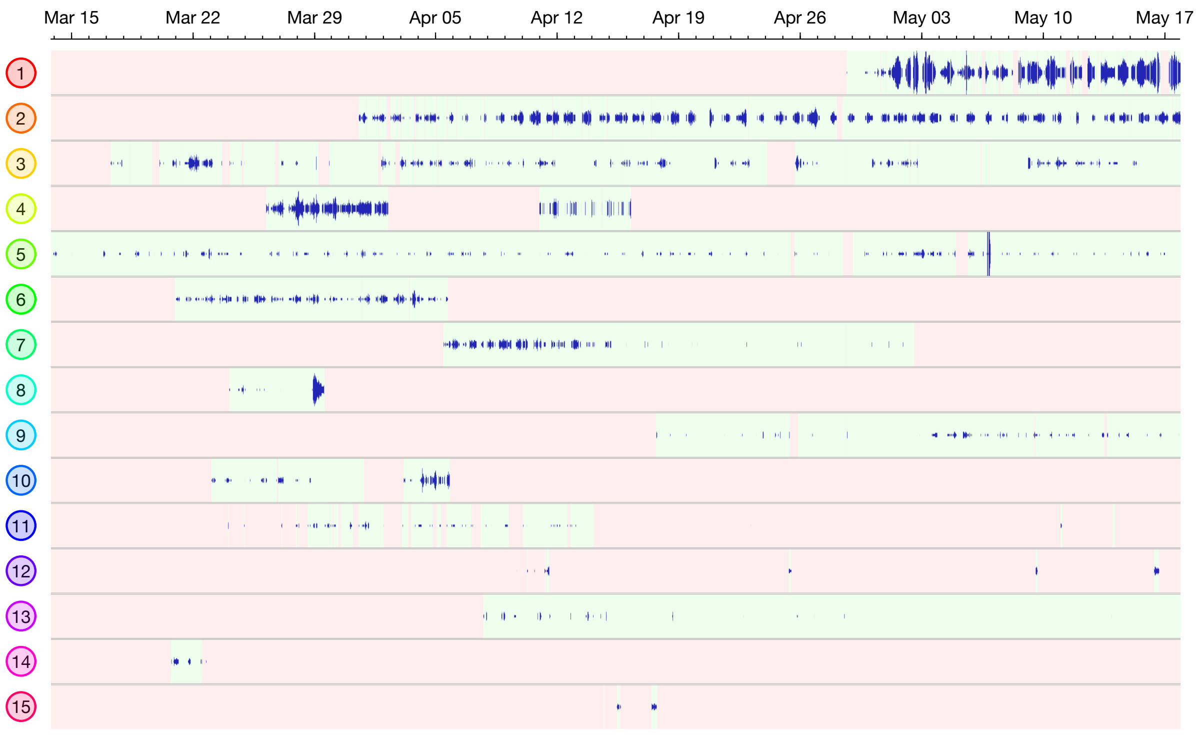


Figure 3.3.: *Amount of active gameplay recorded from different servers over time. The blue lines represent number of active players from 0 to 20. Periods where a server was not reachable are marked red.*

server	players	active	tools
● 1	65	903.36	Classify
● 2	13	865.75	Dashboard
● 3	23	344.01	Classify
● 4	56	273.18	Dashboard
● 5	28	226.68	Dashboard, Classify, DiviningRod
● 6	11	227.13	Dashboard
● 7	7	137.34	Dashboard
● 8	14	95.74	Dashboard
● 9	5	73.35	DiviningRod
● 10	14	54.97	Classify
● 11	3	53.14	Classify
● 12	3	30.05	Classify
● 13	4	24.58	Classify
● 14	4	18.13	Dashboard
● 15	2	14.24	Classify

Table 3.1.: *Number of unique players having more than 1 hour of gameplay, total amount of gameplay, and `HeapCraft` tools used by the different servers.*

attack can be seen as a spike on May 5 on server 1. Server 5 is our own server. We had a LAN party May 6 which is also indicated by a clear spike. Server 12 and server 15 appear to be turned off while nobody is playing. Server 8 and server 14 seem to have reset their server[1] or deinstalled the plugin after a few days.

3.3. Player Classification

We classify a player's average gameplay every 3 minutes, using the classes build, mine, fight, explore and idle. The classifier was developed in our previous project using "experience-sampled" survey data combined with low-level player events [MKF$^+$ar]. The classifier was trained using 3 minute intervals. The Classify plugin can be used to see the classifier in action, where it updates a player's predicted states every three seconds.

The classifier detects typical behavior for each of the 5 classes and quantifies their deviation from average gameplay. It then chooses the behavior which was observed the most during the

[1] After a server is reset, it registers as a different server to us.

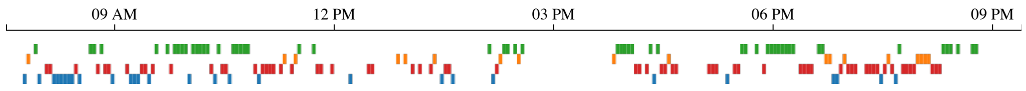


Figure 3.4.: *A player's behavior over one day: mine (green), build (orange), fight (red) and explore (blue).*

3-minute period. The typical behavior of a single player can be seen in Figure 3.4. Players alternate between different behaviors over time.

3.4. Player Contact

While observing collaborating players, we noticed that they often stay within 10 blocks from each other. The distance increases for certain tasks, like building. Hostile players are not considered a serious threat when they are more than 20 blocks away. Players in our dataset are closer than 15 blocks 90% of the time when they are less than 100 blocks away from each other. We justify a contact distance threshold of 15 blocks by those observations.

We define player being in contact with another player after their distance becomes smaller than 15 blocks and both players are active. Contact ends when the distance becomes larger than 20 blocks or either of the players stops being active. The offset is added to avoid fluctuations when players keep at a distance around the threshold.

We recorded a total of 581246 player contacts. Most of them are the result of the alleged denial of service attack mentioned in Section 3.2. Since we are not interested in players not actually playing on the server, we excluded players active for less than 10 minutes. This leaves us with a new total of 99604 player contacts.

Looking at the histogram of contact duration shown in Figure 3.5, there is a frequency spike around 3 seconds. This is about the contact time resulting from walking by a player without interaction. To exclude this and similar meaningless encounters, we ignore contacts lasting less than 10 seconds, removing 33% of all encounters but only 5% of the total contact time.

Minecraft updates players' positions at up to 20 times per second. Since calculating the distance between all players is a relatively computing-intensive task, we limit the contact checks to once per second. This adds quantization errors of up to $\pm$ 1 second to individual contact durations.

Finally, we combine contact periods less than 20 minutes apart. Two players having contact for 5 minutes, no contact for 19 minutes and then contact for 5 minutes again would therefore be counted as having contact for 10 minutes once. If the non-contact phase lasts 21 the encounters would count as two separate contacts lasting 5 minutes each. This transforms the contact frequency metric from indicating how often a player leaves the contact distance threshold to a more meaningful indicator for how often two players interact. The result is 8950 player contacts and an unchanged total contact duration of 45.2 hours.

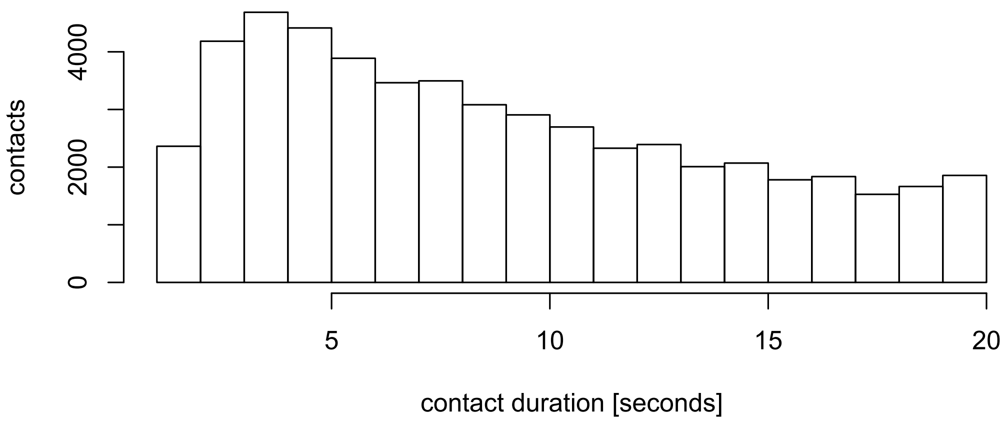


Figure 3.5.: *Histogram of contact durations below 20 seconds for players active for more than 10 minutes.*

3.5. Chat

Minecraft offers a global in-game chat as a convenient communication channel. Players often use this feature to imitate speech by chatting with players nearby, but also to communicate with players far away. The chat feature is sometimes supplemented or replaced with VoIP solutions outside the game.

We recorded 80117 chat messages across all 15 servers. We use a simple model to detect conversations where we assume every message is a reply to all messages by other players which occurred up to 20 seconds before the message. For example, if there are 3 messages by 3 different players within 20 seconds, the second message is a reply to the first message, and the third message is a reply to the first and the second message.

Our reply model contains many false positives by possibly connecting independent conversations. However, players waiting for a reply are likely to read all broadcasted messages which somewhat justifies counting them as being part of simultaneous conversations.

3.6. Buildings

In Minecraft, buildings are created by players assembling previously mined blocks into new structures. There is no clear notion of what constitutes a building, and blocks can be placed for other reasons, like to provide light, stop lava or trap monsters.

We define a building as 20 or more adjacent building blocks[2] placed by players. A block is

[2]See Appendix C.2.1 for a list of blocks categorized as building-relevant.

adjacent to another block if it is within a 3x3x3 cube around the block in question.

To identify buildings, we first compile a list of all building blocks placed by players. We then perform 3D flood fills on that list until all blocks are covered, resulting in 19932 clusters across all 15 servers. 7423 clusters (37%) contain only a single block. 16111 clusters (81%) contain less than 10 blocks. We assume they don't represent buildings but are blocks placed to be markers or small pillars to reach a high place or pass an obstacle. We dropped an additional 1786 clusters by requiring a cluster size of 20 or more. A building consisting of less than 20 blocks is still very small and likely to be a fragment of a larger structure like a wall in front of a natural cave. The remaining 2035 clusters (10%) we recognise as buildings contain 627883 of 693888 placed building blocks (90%).

In order to measure collaborative building, we highlight buildings created by multiple players. From 2035 buildings across all 15 servers, 463 have contributions from more than one player. We decided to ignore players with contributions of less than 5 blocks. This eliminated 10% of all contributions, leaving 301 collaborative buildings with an average of 2.27 contributors. These 301 buildings contain 52% of all placed building blocks.

Since we are limited to the data recorded by the logging plugin, contributions made while logging was inactive are not detected. This results in buildings not showing up in our statistics, and modifications to existing buildings being counted as separate buildings.

3.7. Farms

Edible plants are a commonly used food source in Minecraft. Some of them can be grown by placing seeds on previously prepared soil and harvested about 40 minutes later.

We define farms similarly to buildings but with farm-relevant blocks[3]. The fill algorithm used for detecting farms is more greedy than that we applied to buildings: it searches for neighbor blocks within a 5x5x5 cube. This allows separation of crops by one block within one farm, to reflect common farm designs that provide water or separate crop types. Increasing the size of the search cube further increased farm sizes only minimally.

The clustering happens not only on the currently existing blocks, but on all farm blocks placed and removed on record. So the farm size equals the amount of crops planted and harvested combined. Our algorithm detects 473 farms across all 15 servers with an average of 79.35 and a median of 26 accesses per farm. All clusters have a size of 8 or more, so no further filtering is required.

In order to measure collaborative farming, we highlight farms used by multiple players. From 473 farms across all 15 servers, 184 have been used by more than one player. We decided to ignore farm usage involving less than 5 accesses, eliminating 10% of all usages. This leaves 156 collaborative farms with an average of 2.64 users. These 156 farms cover 69% of all farming activity.

[3]See Appendix C.2.2 for a list of blocks categorized as farm-relevant.

3.8. Chests

In Minecraft, chests can be used to store items. Players store items to free up inventory space or to protect valuable items from being lost in case of death. Chests are also used to share items among players.

For every chest access, we count the number of items a player puts in or takes out. We recorded 4570379 item transfers on 5251 different chests across all 15 servers.

In order to measure resource sharing, we highlight chests used by multiple players. From 5251 chests across all 15 servers, 1407 have been used by more than one player. Shared chests have been accessed by 2.50 on average.

Since we are limited to the data recorded by the logging plugin, contributions made while logging was inactive are not detected. This results in item movements and chests with no access during the recording period not showing up in our statistics.

4

Quantifying Collaboration

The player-driven nature of Minecraft results in a wide variety of goals players might want to achieve. Many of those goals can be reached more efficiently and are more fun to work towards in collaboration with other players. This leads to many different types of collaboration.

A quite obvious type of collaboration is the shared building project. Multiple players might place blocks that ultimately constitute a single structure. In addition to placing blocks, players can help by taking management roles, providing building materials or protecting other players from hostile players and computer-controlled monsters.

Miners and explorers also benefit from working together. Players are able to protect each other from attacks and are able to spot valuable resources more quickly.

Sharing infrastructure like plant and animal farms can be very beneficial to players. The initial costs in time and resources of building a farm are relatively high, whereas maintaining a farm tends to be cheap. Farms also need to be restocked and protected.

Players might also simply seek company. This can be in the form of shared adventures, having conversations or even just trying to kill each other or participate in other competitive activities. It might not look like collaboration but it fits the definition if the shared goal is the enjoyment of the game.

In order to quantify a general conception of collaboration, we started with several more primitive indicators. Our player contact measure may signal collaborative behavior that occurs at the same time and place for both players. Chat, while often used to communicate with players nearby, also indicates remote collaboration of players at different places. Conversely, collaborative building, collaborative farming and shared chests may signal collaborations that are occurring at one location, but possibly asynchronously, with different players active at different times.

True collaborative activity will usually be indicated by several of these primitive indicators. A shared building project for example might include player contact, collaborative building, chat and shared chests.

We use the primitive collaboration indicators to construct undirected, weighted graphs where the vertices V represent players having more than one hour of gameplay and the edges represent collaboration between two players. Using graphs allows us to use terms from network theory to describe the indicator's features. In the visualizations, the players are color-coded by server. The edge widths are represented by their thickness. We use the formula $[1-(1/(w/100+1))]*4$ to only get values between 0 and 4. The edges are transparent, and increased color saturation indicates several edges overlapping.

4.1. Contact Graph

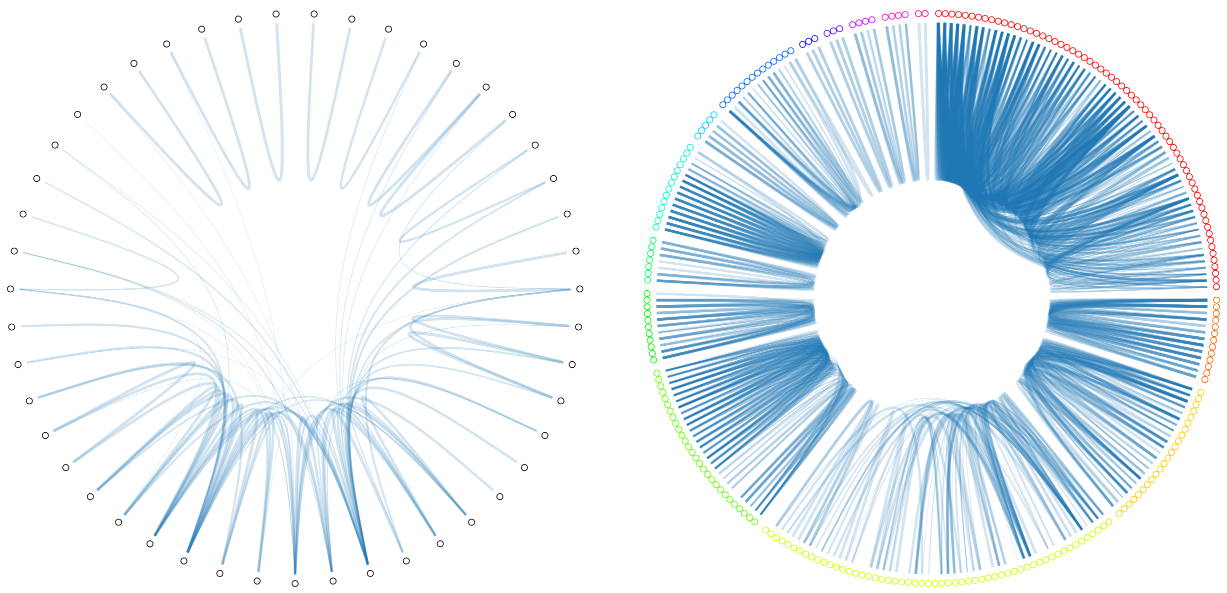


Figure 4.1.: *Contact graph for server 4 (left) and all servers combined (right).*

Player contact is part of almost all of the initially mentioned kinds of collaboration and it is easy to measure and interpret. We define the contact graph $G_{contact} = \langle V_{contact}, E_{contact} \rangle$ where the edges represent contact between two players, weighted by the sum of all contact durations. The resulting graph has 1138 edges across 242 nodes.

The contact graph covers 96% of all players. Its edge weights represent time spent near other active players. Active playtime is considered a valuable resource and is also a stable and fine-grained method of measurement. Since we exclude the time during which any of the players are not active, periods where players leave their computer in order to wait for daytime or wait for plants to grow are excluded. Those periods don't indicate any kind of collaboration and are quite common. Our data shows many instances of players remaining inactive for hours. Player contact is part of almost all of the initially mentioned kinds of collaboration and it is easy to measure and interpret.

4.2. Chat Graph

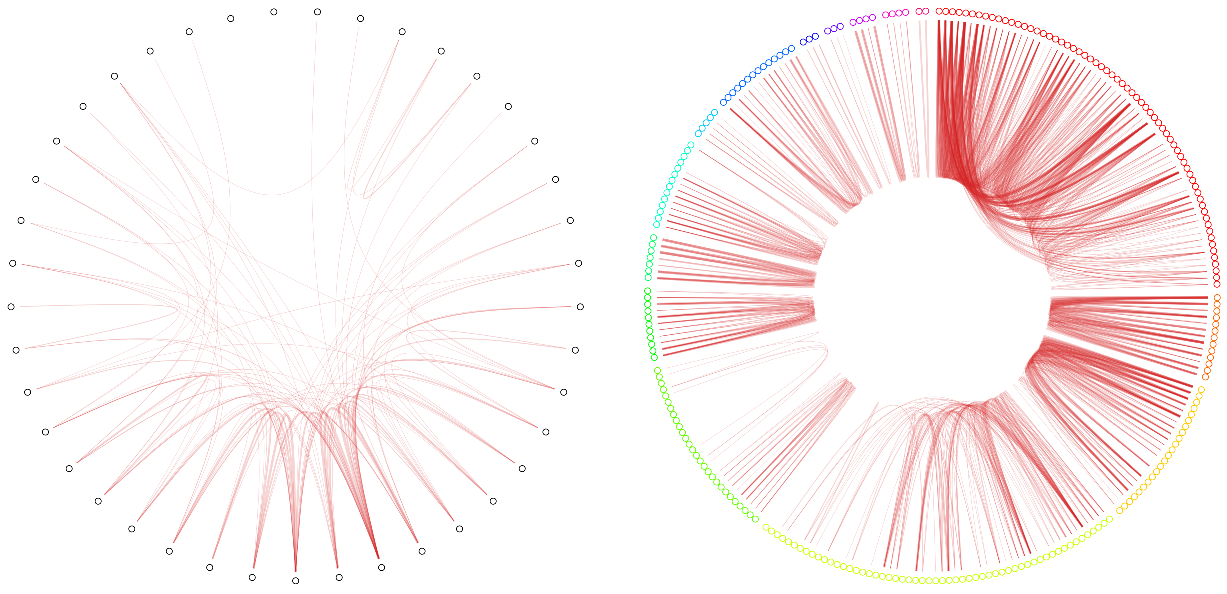


Figure 4.2.: *Chat graph for server 4 (left) and all servers combined (right).*

The chat graph $G_{chat} = \langle V_{chat}, E_{chat} \rangle$ connects players who have participated in a shared conversation. The weight of the edges represents the sum of answers given or received between two players. The resulting graph has 1008 edges across 221 nodes.

Of course, conversation does not imply collaboration, but combined with other noisy measures of collaboration, the chat graph can contribute to a characterization of general collaboration.

The chat graph covers 88% of all players. Its edge weights represent number of messages. Naturally, the number of messages sent varies greatly among players. Two players exchanging 100 messages don't necessarily have a deeper relationship than players only exchanging 10. Some players also use alternative ways of communications, like VoIP, which we can't measure at all.

4.3. Building Graph

The building graph $G_{build} = \langle V_{build}, E_{build} \rangle$ connects players who have contributed to the same building. Each collaborative building creates a complete graph connecting all contributors. The weight of the edges represents the sum of contributions two players have made to the same buildings. The resulting graph has 378 edges across 191 nodes.

The build graph covers 76% of all players. Its edge weights represent the number of placed blocks. The meaning of placing a certain amount of blocks changes in different contexts. A simple wall made of hundreds of blocks can be created within a couple of minutes, whereas building the interior of a house can take hours and only involve placing a few blocks. Due to

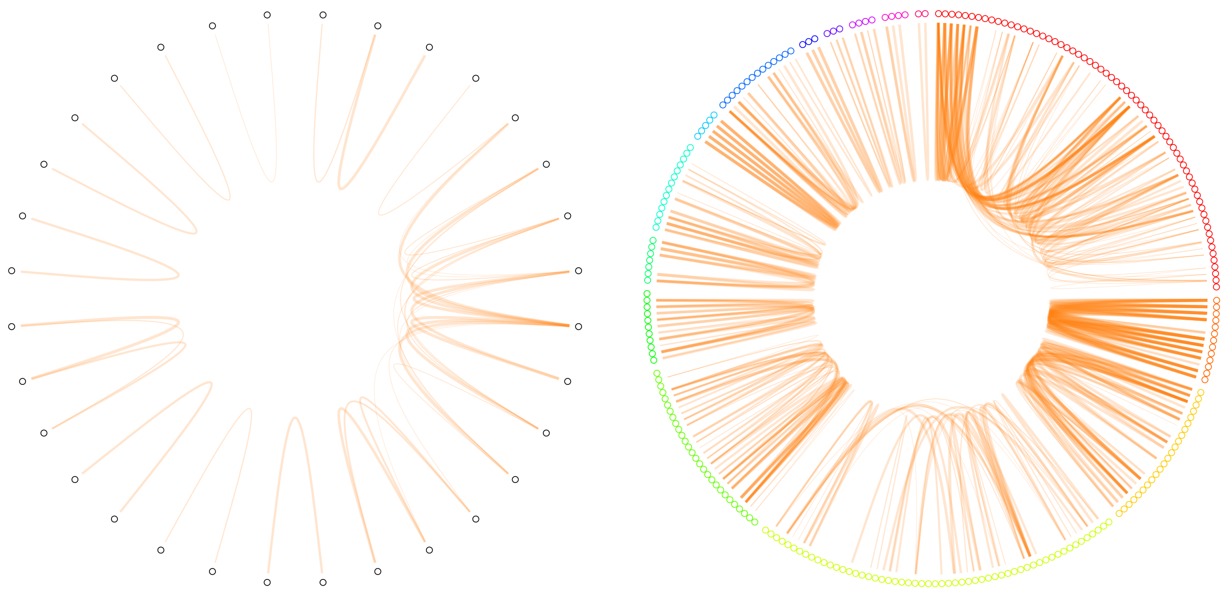


Figure 4.3.: *Build graph for server 4 (left) and all servers combined (right).*

the way we assign placed blocks to all players involved in the building project, large building projects dominate the final edge weight to a probably unjustifiable degree.

4.4. Farm Graph

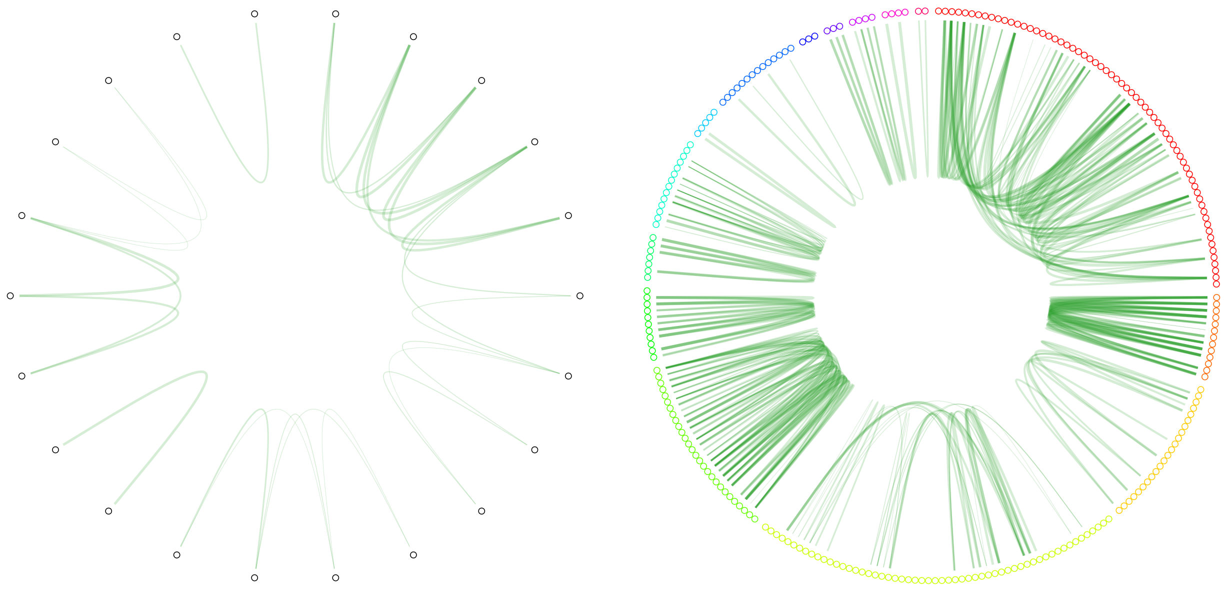


Figure 4.4.: *Farm graph for server 4 (left) and all servers combined (right).*

The farm graph $G_{farm} = \langle V_{farm}, E_{farm} \rangle$ is constructed analogously to the building graph. The weight of the edges represent the sum of farm accesses (plant or harvest) of two players to a shared farm. The resulting graph has 323 edges across 149 nodes.

The farm graph covers 59% of all players. Its edge weights represent accesses to a shared farm (planting or farming a block). The metric has the same problems as the build graph: large farms dominate the edge weights. But since the amount of blocks which can be planted is limited to the farm size and planted blocks need time to grow, the number of farm accesses behaves quite well.

4.5. Chest Graph

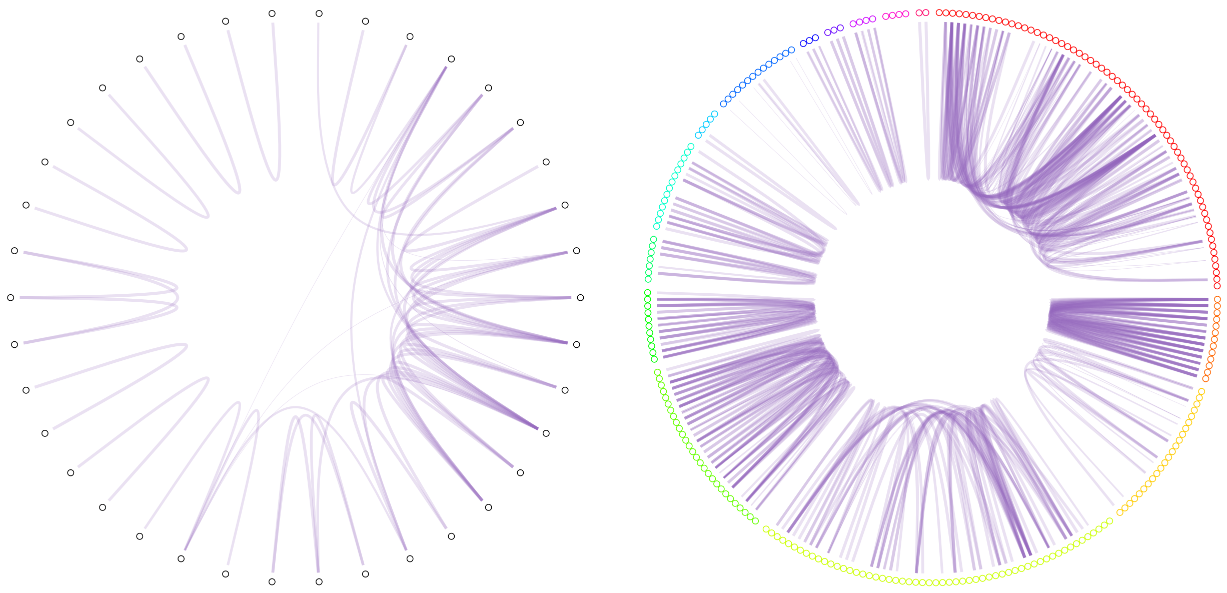


Figure 4.5.: *Chest graph for server 4 (left) and all servers combined (right).*

The chest graph $G_{chest} = \langle V_{chest}, E_{chest} \rangle$ is constructed analogously to the building graph. The weight of the edges represent the sum of transferred items of two players to a shared chest. The resulting graph has 435 edges across 188 nodes.

The chest graph covers 75% of all players. Its edge weights represent number of blocks or other items. Certain types of blocks are nearly worthless and very easy to come by. Putting them in or taking them out of a chest takes very little effort and is often done with hundreds at a time. While mining, getting rid of unwanted blocks by dropping them off in random chests is very common.

4.6. Graph Comparison

Take Away Points The five graph types define edges between many of the same pairs of players. Their similarity indicates a correlation between different kinds of collaboration, and it hints at

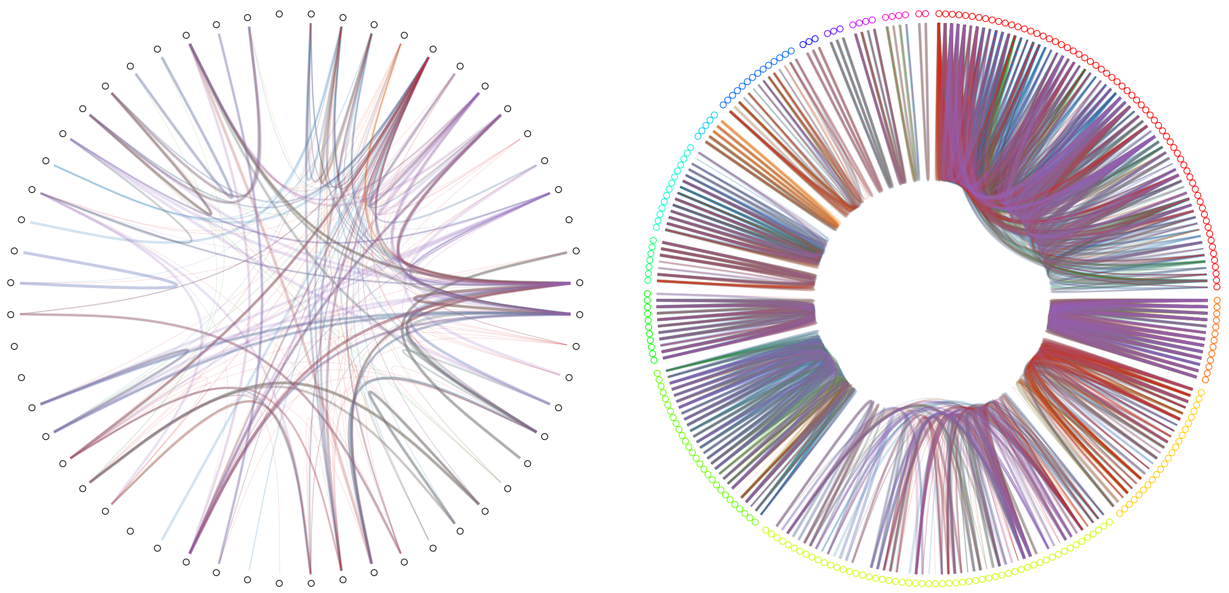


Figure 4.6.: Combined graph for server 4 (left) and all servers combined (right). contact (blue), build (orange), farm (green), chat (red), chest (purple)

the possibility of defining one generic metric for collaboration. In particular, the contact graph contains large parts of every other graph and is therefore a good candidate to be used for that metric.

No players in our dataset were active on more than one server, so each server adds a subgraph of unconnected players to the combined graph. Table 4.1 shows the player coverages of those subgraphs on each server in case of the contact graph. The player coverage is calculated by dividing the number of edges by the number of edges of a completely connected graph. A coverage of 0.2 means a player is connected to 20% of all other players in average.

The average edge is inside a subgraph containing 47 players. The average player is in a subgraph containing 212 edges. We use this information to model a typical player connection graph. If we take any graph with 47 vertices and 212 edges and compare it with a similar graph with 212 randomly inserted edges, $212/1081 * 212 = 41.58$ edges will be in both graphs on average. So two player connection graphs with independent edges are expected to share 20% of their edges. The number will be smaller when comparing graphs of smaller sizes, but we assume values over 20% indicate two graphs representing similar things.

We use the function

$$\mathrm{cover}(G_1, G_2) = |E(G_1) \cap E(G_2)|/|E(G_2)|$$

to express how much of graph G_1 covers G_2 or in other words, how many of the edges of G_2 are also present in G_1. Figure 4.7 shows the result of the function for all previously defined graphs. Note that the contact graph contains at least 70% of edges from all other graphs. Since the coverages are all much higher than the 20% expected by the typical graph, all 5 graphs must

server	players	edges	coverage
● 1	65	617	0.30
● 2	13	47	0.60
● 3	23	80	0.32
● 4	56	86	0.06
● 5	28	159	0.42
● 6	11	29	0.53
● 7	7	12	0.57
● 8	14	58	0.64
● 9	5	7	0.70
● 10	14	25	0.27
● 11	3	3	1.00
● 12	3	3	1.00
● 13	4	6	1.00
● 14	4	5	0.83
● 15	2	1	1.00

Table 4.1.: *Player coverage of the contact graph on different servers.*

be strong indicators of a shared hidden variable. Since the graphs are designed to represent different aspects of collaboration, we believe the hidden variable to indicate collaboration as well.

4.7. Collaboration Index

In order to quantify collaboration, we not only need to know about player relationships, but also how to quantify them. The high coverage of the contact graph and the appealing properties of its edge weights make contact time an excellent candidate for quantifying collaboration. Player contact is part of almost all of the initially mentioned kinds of collaboration and it is easy to measure and interpret. Since the contact graph contains large parts of all other graphs, and since we desire a measure that is easy to calculate, we do not explicitly incorporate the other collaboration metrics into our general metric. Instead, we use their overlaps with the contact graph to justify our decision to define collaboration in terms of player contacts exclusively.

We introduce the collaboration index I_c as an universal indicator for collaboration for players and servers:

graph	vertices	edges	players without edges
contact	242	1138	10 (4%)
chat	221	1008	31 (12%)
build	191	378	61 (24%)
farm	149	323	103 (41%)
chest	188	435	64 (25%)

Table 4.2.: *Number of vertices and edges for the different graphs.*

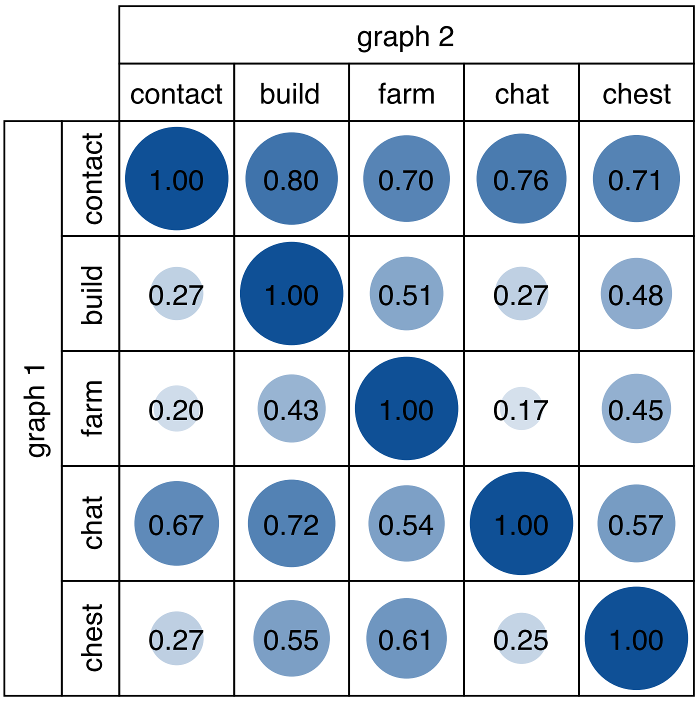


Figure 4.7.: *Graph coverage matrix: How many edges of graph 2 are included in graph 1?*

$$I_c(p) = \frac{\sum_{e \in E_{contact}(p)} \mathrm{W}[e]}{\mathrm{active}(p)} \qquad \text{collaboration index of a player}$$

$$I_c(s) = \frac{\sum_{e \in E_{contact}(s)} \mathrm{W}[e]}{\mathrm{active}(s)} \qquad \text{collaboration index of a server}$$

where $E_{contact}(\cdot)$ represents the set of all contact graph edges of a player or server, $\mathrm{W}[\cdot]$ the weight of an edge and $\mathrm{active}(\cdot)$ the duration of active gameplay of a player or server.

The collaboration index can be interpreted as a player's or server's average number of simultaneous player contacts while those players are active. We chose to define I_c as normalized over the total amount of active gameplay to allow comparisons between different players and servers.

5 Predicting Collaboration

5.1. Player Analysis

Take Away Points Our data indicates that players prefer to meet players they already know. Collaborative players meet more other players and/or meet them for longer, but not more frequently than do average players. Collaboration shows strong positive correlations with fighting and negative correlation with mining.

We calculate the following variables for all 252 players across the 15 servers:

- $\mathbf{active}$: active gameplay in hours
- $\mathbf{collab}$: collaboration index (average number of simultan player contacts while active)
- $\mathbf{nContact}$: number of different players a player has had contact with; indicates player popularity
- $\mathbf{fContact}$: average number of contacts with a specific player (frequency)
- $\mathbf{tContact}$: average contact duration; indicates player familiarity
- $\mathbf{chestContrib}$: shared chest $in/(in+out)$ from -1 (only taking items) to 1 (only putting items in)
- $\mathbf{farmContrib}$: shared farm $in/(in+out)$ from -1 (only farming) to 1 (only planting)
- $\mathbf{farm}$: average farm usage $(in+out)/active$
- $\mathbf{chat}$: average number of chat messages sent $nMsg/\mathbf{active}$
- $\mathbf{build}$, $\mathbf{mine}$, $\mathbf{farm}$, $\mathbf{explore}$: percentage of 3 minute time slices classified as the given

	active	collab	nContact	fContact	tContact	farm	chat	b	m	f	e	special
mean	13.26	0.32	10.15	2.95	0.12	41.91	23.41	0.26	0.13	0.23	0.38	0.04
sd	23.26	0.26	11.46	3.05	0.12	102.41	32.95	0.16	0.12	0.19	0.19	0.03

Figure 5.1.: *Mean and standard deviation across all players.*

behavior (sum up to 1)

- $\mathbf{special}$: behavior specialization $\mathrm{Var}(\{\mathbf{b}, \mathbf{m}, \mathbf{f}, \mathbf{e}\})$ from 0 (all behaviors are equally frequent) to 0.25 (only one behavior).

In order to compare players with different amounts of active gameplay, some variables are normalized over active gameplay to avoid trivial correlations with $\mathbf{active}$. For example, instead of using the number of chat messages, we calculate the number of chat messages per hour, creating a measure of "player chattiness." Due to the high number of players not using shared chests (59) or shared farms (107), the variables $\mathbf{chestContrib}$ and $\mathbf{farmContrib}$ are discussed separately in the next section.

Figure 5.1 shows the mean and standard deviation across all 252 players and Figure 5.2 their correlation. The correlation matrix includes the correlation between all values so they can all be found in one place, but it includes different independent groups of analyses. Each group of analyses we performed is framed by a green line.

We exclude numeric values from the correlation matrix to improve readability. The raw correlations and their p-values can be found in Appendix C.3. We controlled our number of comparisons (and potential false positives) by attending only to particularly relevant subsets of correlations. Additionally, we adjusted for the number of statistical tests by moving the standard significance threshold of $p < 0.05$ down to $p < 0.01$.

Some of the variables have inherent correlations due to their definition. For example $\mathbf{collab} = \mathbf{nContact} * \mathbf{fContact} * \mathbf{tContact} / \mathbf{active}$. The constraint $\mathbf{b} + \mathbf{m} + \mathbf{f} + \mathbf{e} = 1$ leads to mostly negative correlations between the classification variables.

A correlation between $\mathbf{active}$ and $\mathbf{fContact}$ is expected since the number of encounters with the same player increases over time, even if those encounters are random. But the combination with the much lower correlation to $\mathbf{nContact}$ suggests that players do indeed prefer to meet players they already know. The rest of the correlations with $\mathbf{active}$ are small which is expected due to our normalization. Builders seem to spend slightly more time playing, fighters slightly less. The negative correlation between $\mathbf{active}$ and $\mathbf{collab}$ could be an artefact of normalizing $\mathbf{collab}$ over time, but is more likely caused by the negative correlations between $\mathbf{b}$ and $\mathbf{collab}$, as well as those between $\mathbf{f}$ and $\mathbf{collab}$. The negative correlation between $\mathbf{active}$ and $\mathbf{special}$ indicates that players who spend a lot of time playing tend to have more diverse gameplay.

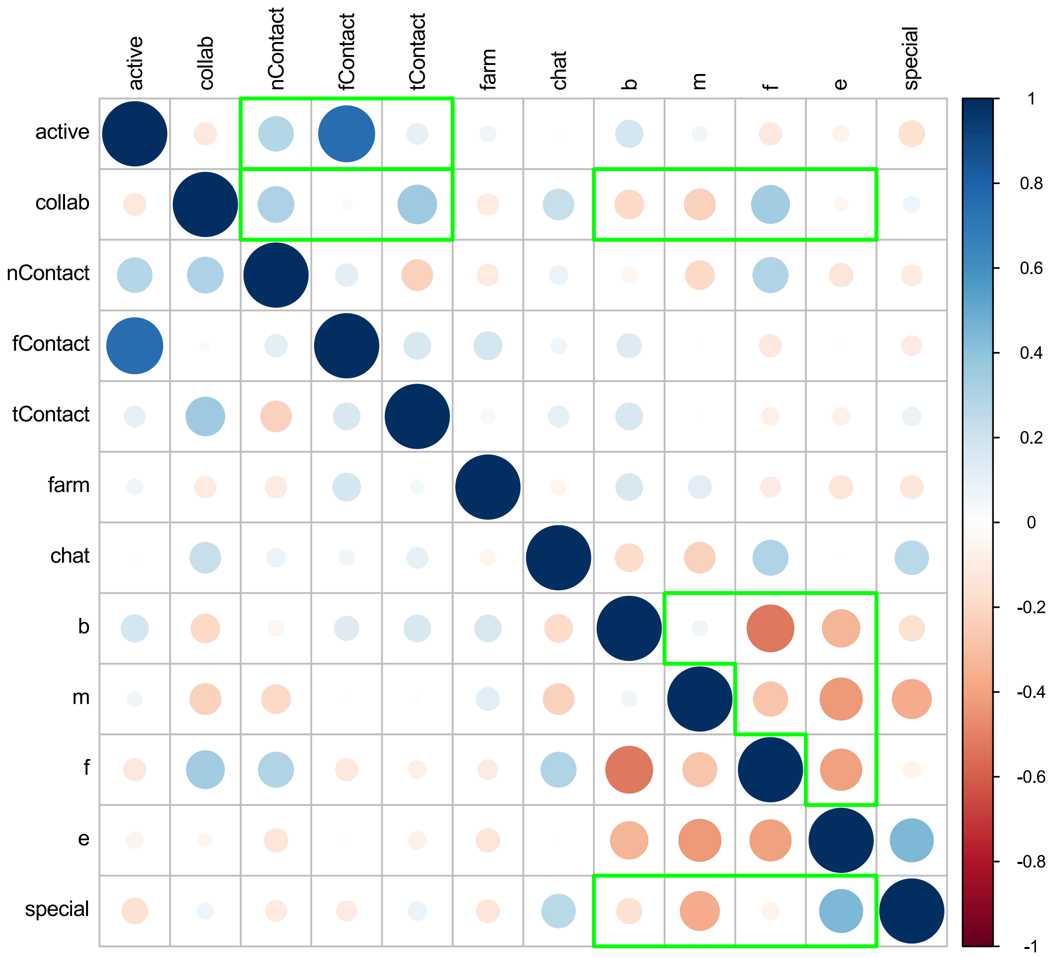


Figure 5.2.: *Correlations between player variables. Specific groups of analyses of interest are bounded in green.*

Since $\mathbf{collab}$ is the product of $\mathbf{nContact}$, $\mathbf{fContact}$ and $\mathbf{tContact}$, comparing the values of their respective correlations is interesting. $\mathbf{Collab}$ vs. $\mathbf{nContact}$ and $\mathbf{collab}$ vs. $\mathbf{tContact}$ show high correlations whereas $\mathbf{collab}$ vs. $\mathbf{fContact}$ shows no significant correlation. This means collaborative players meet more other players and/or meet them for longer, but not more frequently than average players.

$\mathbf{collab}$ shows strong positive correlations with fighting and negative correlation with mining. The slightly lower negative correlation with building has a p-value over 0.01. It should be noted that the classified behaviors don't represent collaborative building, mining and fighting, but what the player is doing alone and with others combined. The positive correlation with $\mathbf{chat}$ supports our argument of the collaboration index being a general indicator that can detect different kinds of collaboration.

Since $\mathbf{special}$ represents variance of the behavior variables, its average value of 0.04 equals a standard deviation of $\sqrt{0.04} = 0.19$ between them. So the different behaviors are expected to deviate from their mean of 0.5 by about 0.19 on average. This means all 4 behaviors tend to be well-represented by each player while there are still clearly visible preferences. Explorers tend to be more specialized while miners tend to be less specialized.

The inherent connection of the classification variables adds a negative bias to their correlations. Most negative correlations can be ignored. On the other hand, the positive correlation between building and mining is more significant than it seems. The correlation can easily be explained with the observation that obtaining blocks by mining and using them for building go hand in hand. The two tasks could be assigned to different players, but players specialized on mining seem rare and players who build and mine also seem to be less collaborative. The strong negative correlation between building and fighting indicate players tending to prefer either one or the other.

5.2. Contribution to Common Goods

Take Away Points Collaborative players tend to be more socially responsible by giving more to shared resources than they take.

By default, players who develop resources like productive farms or well stocked chests can do little-to-nothing to prevent others from benefitting from them. This scenario incentivizes players to extract from common goods without contributing to their maintenance. So players who continue to contribute to common goods are exhibiting prosocial behavior. The granularity of our data allows us to detect instances of contributing to the sundry common goods that implicitly emerge in Minecraft gameplay.

We analyzed two types of common goods: Chests accessed by more than one player and farms accessed by more than one player. Our dataset contains 193 players with shared chests and 184 players with shared farms. We measured contribution by counting items put into and taken out of a chest or plants planted on and harvested from farms. The variable

$$\mathbf{contrib} = 2 * in/(in + out) - 1$$

assigns a value from -1 to 1 to every player indicating how much they remove from or add to a given type of resource. A value of 0 means they take as much as they give; a value of -1 means they only take (freeloader); a value of 1 means they only give (benefactor). Players not using any of the shared resources are assigned a 0.

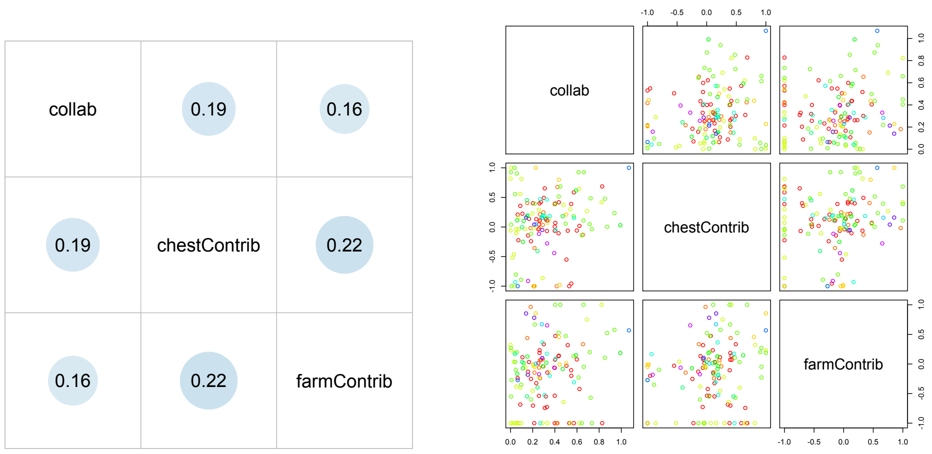


Figure 5.3.: *Correlations between collaboration and level of contribution to shared chests and farms.*

Since contributions to farms and chests will not correlate with other variables if a player is not using shared resources, we exclude them when calculating the correlation matrix. Among the 134 players using both shared chests and shared farms, there is relatively strong correlation between both contribution types $(r = 0.22, \quad p = 0.01)$. Players have been observed storing what they farmed in chests and taking seeds to plant from chests, which would suggest a negative correlation. Since there is no other obvious connection between farming and chest usage, we speculate that the positive correlation is caused by a hidden linked variable indicating a player's sense for social responsibility.

The strong correlation between **collab** and contribution is impressive since there is no easy way for a player to identify another player as a benefactor or a freeloader. However, the decreased sample size by excluding players who don't use shared resources limits the significance of the correlations (p=0.05).

5.3. Player Pair Analysis

Take Away Points Players doing much fighting but little building tend to team up with their opposites. Builders tend to team up with other builders and fighters with other fighters. Players specialized on exploring tend to play with more balance players.

We looked at the average and difference of player variables for each edge of the contact graph using the following metrics:

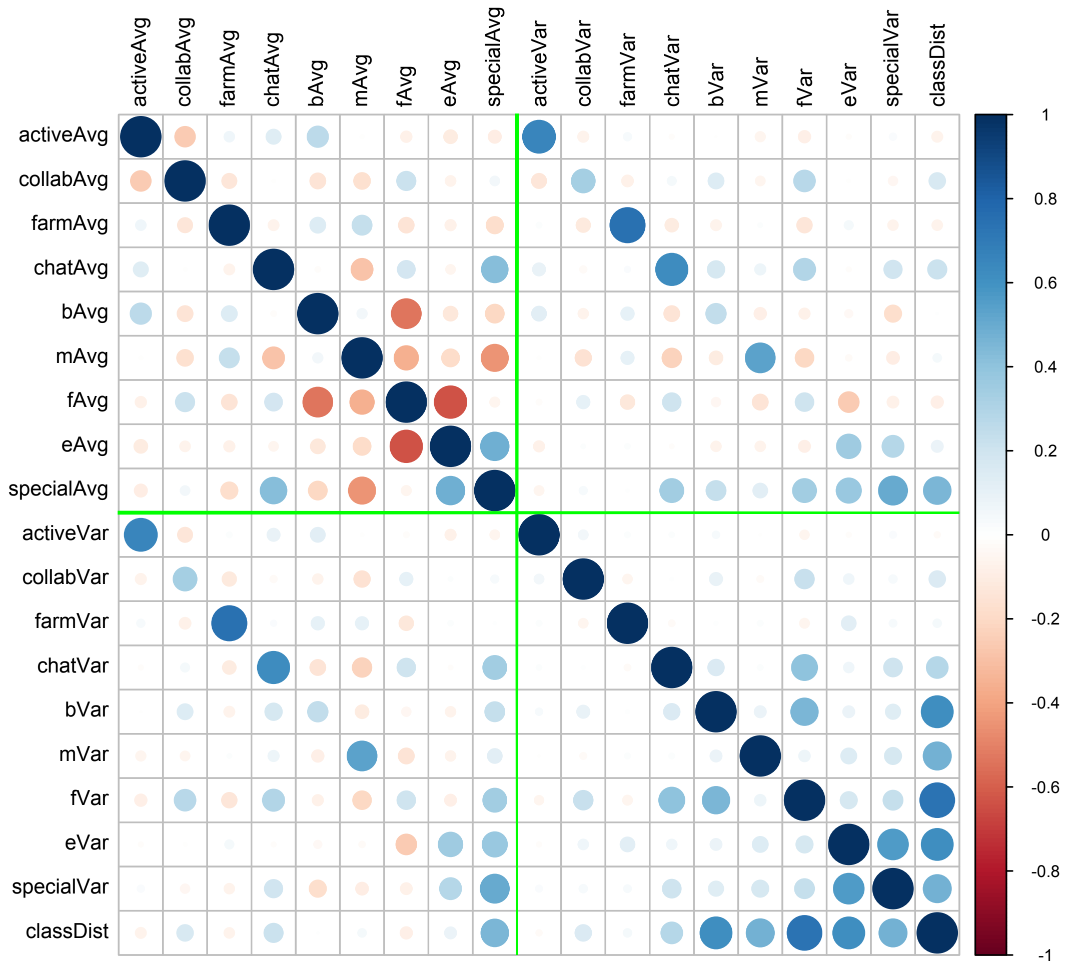


Figure 5.4.: *Correlation between player contact pairs. For each basic variable, the average and variance between the two players are used separately.*

$$
\begin{aligned}
\mathbf{vAvg} &= (v_{p1} + v_{p2})/2 \\
\mathbf{vVar} &= \mathrm{Var}(\{v_{p1}, v_{p2}\}) = (v_{p1} - v_{p2})^2/4 \\
\mathbf{classDist} &= [(b_{p1} - b_{p2})^2 + (m_{p1} - m_{p2})^2 + (f_{p1} - f_{p2})^2 + (e_{p1} - e_{p2})^2]/4
\end{aligned}
$$

vAvg represents the average value of the variable **v** of two players and **vVar** their difference expressed by variance. **classDist** can be interpreted as the mean square error whereas the variance of two variables equals halve their mean square error. The relationship $b+m+f+1 = 1$ limits the range of **classDist** to values from 0 (same behavior) to 0.5 (full specialization on two different behaviors).

The top left quarter is expected to show similar results to the per player correlation matrix in Figure 5.4, but with more weight on players with high **nContact** (contacts with many different players). The lower left quarter shows correlations between variable differences among player pairs. The remaining two quarters show correlations between the pair average and the pair differences. Most correlations in the remaining two quarters can be explained by correlations already observed in the per player matrix and the fact that high averages often cause high variances. The higher sample size of 1138 (number of edges in the contact graph) might lead to the impression that the significance of the results are higher than the results of the per player analysis. However, the samples are still based on the same 252 players. For this reason, conclusions in this section are only of qualitative nature.

The main differences between the per pair average and the per player correlation matrix are a stronger negative correlation between $\mathbf{active}$ and $\mathbf{collab}$, a positive correlation between $\mathbf{chat}$ and $\mathbf{active}$ instead of $\mathbf{chat}$ and $\mathbf{collab}$ and weaker negative correlations between $\mathbf{explore}$ vs. $\mathbf{build}$ and $\mathbf{explore}$ vs. $\mathbf{mine}$, but a stronger negative correlation between $\mathbf{explore}$ and $\mathbf{fight}$. Interpretation of these differences is difficult without much deeper analysis since they don't seem to be directly related to players with higher $\mathbf{nContact}$.

In the bottom right quarter, a strong correlation between $\mathbf{bVar}$ and $\mathbf{fVar}$ can be found, indicating that players doing much fighting but little building tend to team up with their opposites. The correlation between $\mathbf{chatVar}$ and $\mathbf{fVar}$ can be explained by the correlation between $\mathbf{fight}$ and $\mathbf{chat}$.

The correlations between the averages and the variance among the different classifications are significantly different: build: 0.25, mine: 0.54, fight: 0.2, explore: 0.36. Those differences stay similar when removing the normalization by their standard deviation and are therefore not a result of inherently different properties of the variables in respect to mean and variance. Since variance is a function of the mean, high values are expected. Standing out are building and fighting with very low values. They indicate that builders tend to team up with other builders and fighters with other fighters. Shared interests in mining on the other hand seems to decrease the likeliness of teaming up.

A strong correlation between $\mathbf{specialVar}$ and $\mathbf{eVar}$ suggests that players specialized on exploring tend to play with more balance players.

5.3.1. Entropy

Take Away Points Players prefer to team up with players with similar preferences in building, mining, fighting and exploring whereas differences in experience are irrelevant.

We calculate whether the player contact pairings increase or decrease the variable differences compared to random pairings. We use the formula $dQ = (\mathrm{E}[vVar] - u)/sd$ where u represents the mean variance across all possible player pairs. The difference is then divided by the standard deviation sd calculated across all possible player pairs in order to make the resulting variables comparable. dQ can be interpreted as the difference in entropy the player pairings lead to.

The results in Figure 5.5 indicate that players prefer to team up with similar players, especially in respect to fighting and exploring. The difference between active seems to be not affected by the pairings. This indicates the irrelevance of differences in experience level for collaboration.

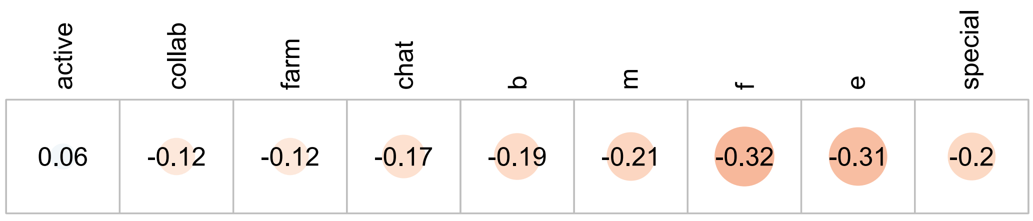


Figure 5.5.: Average change in entropy dQ resulting from player contact pairs compared to random pairings.

5.4. Simultaneously Active Players

Take Away Points Playing at peak times increases collaboration.

We calculate the average number of active players on a server while a particular player is active ($\mathbf{nActive}$) and compare it with the collaboration index of that player. A player who is the only active player for one hour would have a $\mathbf{nActive}$ value of 1. After playing another hour while 3 other players are active, the value would increase to a new average of $1 * 1 + 1 * 4 = 2.5$. Since $\mathbf{nActive}$ is limited by the total amount of different players on a server, we risk detecting properties of different servers instead of generalizable player characteristics when looking at correlations with $\mathbf{nActive}$. Table Table 5.1 suggests that the strong correlation between $\mathbf{nActive}$ and $\mathbf{collab}$ is valid across all servers, unless they have few players.

The extreme values for servers with few players can be explained by the high impact of individual players. The green cloud at the right edge in Figure 5.6 is the result of a LAN party we organized on our own server.

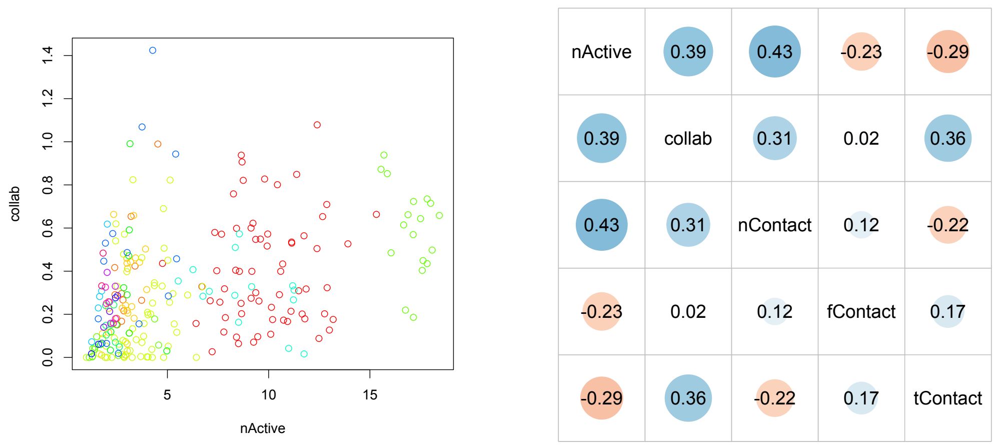


Figure 5.6.: The correlation between the average numbers of active players while a player is active ($\mathbf{nActive}$) and a players collaboration index ($\mathbf{collab}$). Players are colored by server.

The strong correlation between $\mathbf{nActive}$ and $\mathbf{collab}$ $(r = 0.39, \quad p < 0.001)$ indicates that

server	players	average $\mathbf{collab}$	average $\mathbf{nActive}$	$\mathrm{Cor}(\mathbf{collab}, \mathbf{nActive})$
● 1	65	4.89	9.91	0.70
● 2	13	0.48	2.92	0.74
● 3	23	0.32	2.96	0.49
● 4	56	0.52	3.66	0.35
● 5	28	1.38	11.74	0.80
● 6	11	0.34	2.90	0.06
● 7	7	0.26	2.22	0.45
● 8	14	0.67	8.92	−0.59
● 9	5	0.29	1.66	0.97
● 10	14	0.46	3.19	0.47
● 11	3	0.11	1.76	0.99
● 12	3	0.28	2.18	0.99
● 13	4	0.37	2.20	−0.41
● 14	4	0.71	2.17	0.91
● 15	2	0.40	1.79	1.00

Table 5.1.: *Number of players, average of per player collaboration index ($\mathbf{collab}$), average of per player average number of players active at the same time ($\mathbf{nActive}$) and correlation between $\mathbf{collab}$ and $\mathbf{nActive}$ for each server.*

playing at peak times increases collaboration. Inspection of player distribution show that players tend to be well distributed over the map either alone or in small groups. Assuming players are randomly distributed over an area of 200 by 200 blocks, the probability of two players having contact is only 0.56%. Players tend to be distributed over even bigger areas and there are usually significantly less than 20 players active at the same time, making an increase in collaboration for purely mechanical reasons unlikely. We believe that players seek or avoid contact consciously and a greater choice in players increases the likelihood of finding a good match.

The correlation matrix shown in Figure 5.6 indicates that when a server has many players active at the same time, most contacts are short and occur only once. Those encounters add little to our metric for collaboration which mainly consists of long repeated contacts. Since we ignore contacts lasting less than 10 seconds, the increase in encounters is not just caused by the higher density of players. The correlation could be the result of by popular places visited often but for short periods of time, like the spawn area.

We speculate that collaboration increases during peak hours due to the increased likelihood of meeting new players and availability of already known players. Newly formed relationships might lead to additional collaboration in the future, even at times with fewer active players.

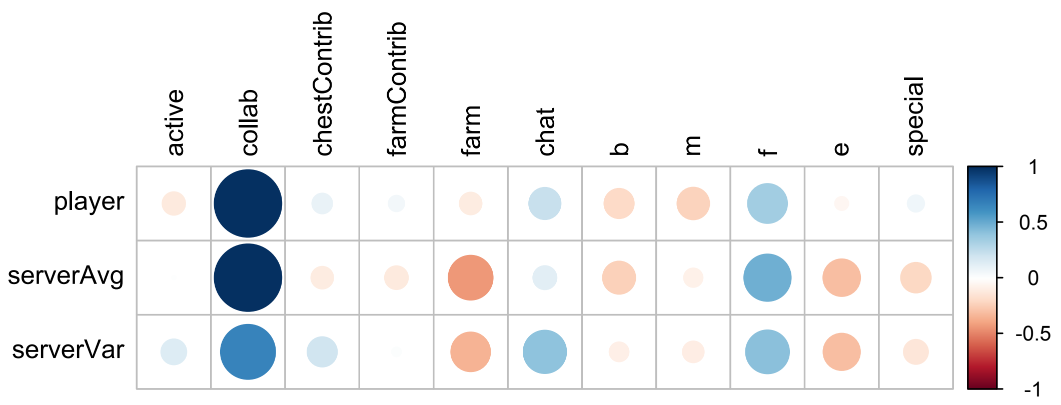


Figure 5.7.: *Correlations of* **collab** *with variables by player, server average and server variance. With only 15 observations, none of these correlations is significant.*

5.5. Server Analysis

We aggregate the variables of the 252 players by server and calculate their correlation with each server's average collaboration index. We used two different methods of aggregation: mean (**serverAvg**) and variance (**serverVar**). Figure 5.7 shows those correlations in addition to the correlations of the non-aggregated player list for comparison.

The low sample size of only 15 servers limit this server-based correlation analysis. For example, even though players with high contribution values tend to be more collaborative, servers with a high average of contributions seem to be less so. The sample size is not large enough to make any of the shown correlations significant.

Collaborative servers show a high variance in collaboration of individual players. They also exhibit less specialization of individual players and have more fighters but fewer builders and explorers and a high variance in chat frequency. However, the small sample size of only 15 servers leaves the expected accuracy of predictions for collaboration on a server by server basis very low and all our findings are statistically insignificant. Most correlations have a p-values of over 0.5.

6

Improving Collaboration

This section contains actionable ideas on how to improve collaboration on a server based on correlational insights derived from our data analysis. Furthermore, it introduce extensions to `HeapCraft` that currently implement each of these suggestions. Experimental evaluation and verification of these claims is outside the scope of this work but, integrated with Epilog's data collection, future work may be able to experimentally test the effectiveness of these extensions.

6.1. DiviningRod and other extensions to HeapCraft

In order to actively influence player collaboration, we created DiviningRod, a Minecraft plugin which makes it easier to find groups or individual players. DiviningRod adds a wide variety of different compasses to Minecraft. They can point to players, specific locations or to player-created signs containing hashtags by displaying their distance from the player's current location as shown in Figure 6.1.

Where a compass points to can be defined on a per-player basis using real-time heuristics provided by our logging server. Researchers can implement their own heuristics and evaluate their effect on players. We collect usage statistics on what kind of compasses are used when, for how long and by whom.

We tested DiviningRod by installing it on our own server. The most popular features include locating the nearest player, a specific player by username or finding a server's spawn point. A more detailed description can be found in the user manual in Appendix B.1.

Other tools that we developed also implement suggestions in this section. Both the Epilog Dashboard and the Classify plugin provide information that, according to our analyses, may

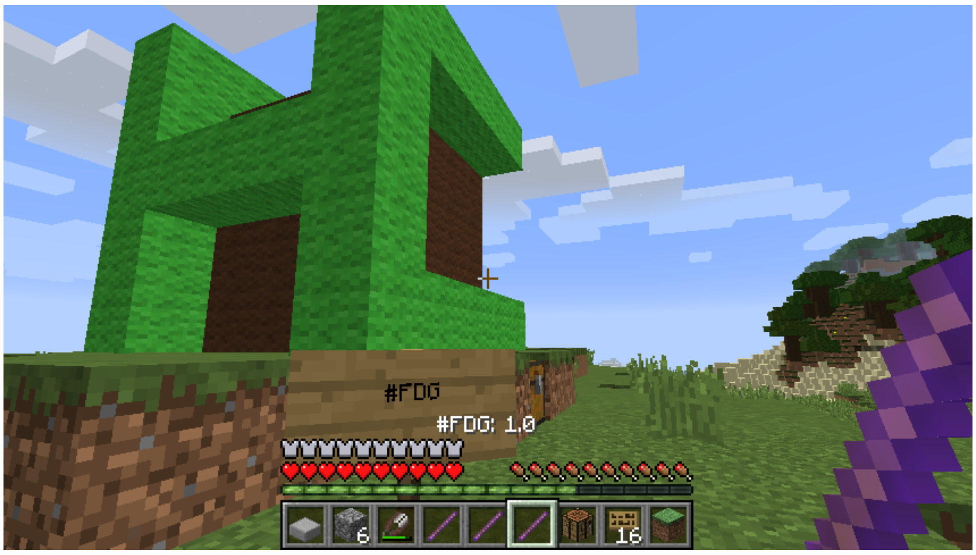


Figure 6.1.: *A compass provided by DiviningRod showing the distance (1 meter) to a sign.*

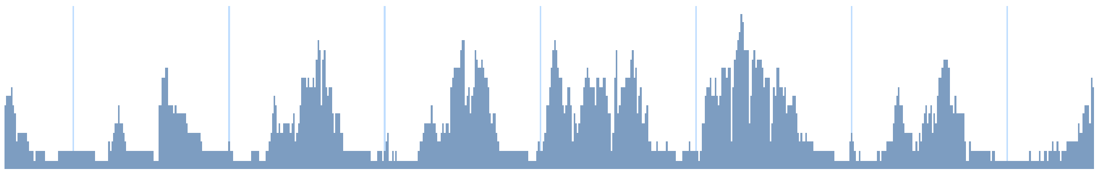


Figure 6.2.: *History of online players on a server as presented in our Epilog Dashboard. The peak represents 22 players. Vertical lines mark 6am local time. The right edge represents now.*

facilitate collaboration on a server.

6.2. Promote Simultaneous Activity

Our data highlights the importance of many players being online at the same time. Visualizations of showing the history of the number of active or online players over time like shown in Figure 6.2 can be used by players to predict good times to play. The Epilog Dashboard includes this graphic which updates in realtime. Administrators can include it in their own website. We also offer the data used to generate the graphic, so administrators are able to create their own graphical representations.

Announcing special events like LAN parties or building projects in advance will also lead to high numbers of players active simultaneously.

6.3. Facilitate Fighting

Our data indicate strong correlations between fighting and collaboration. We have anecdotal evidence that some experienced players prefer challenging fights. Increasing a server's difficulty level will make fighting computer-generated monsters harder and could attract more fighters.

DiviningRod has shown to be a useful tool for "player vs. player" fighting. We observed players using it to observe the distance to their nearest player to either avoid them, prepare themselves for a fight or choose the next player to attack. The ability to locate specific players has also been used for similar purposes. Those new possibilities can make a server more appealing to fighters.

6.4. Improve Player Matching

Our analysis of player collaboration pairs show that some players are more likely to team up than others. Making it easier for players to find a good partner could improve collaboration.

Classify shows whether a player is currently building, mining, fighting, exploring or idle for each online player. This information can be used to find teammates.

DiviningRod can help players to find specific players they want to collaborate with or choose a player that is nearby. DiviningRod also supports finding players based on real-time analytics. This enables the implementation of more complex heuristics based on findings in our player pair analysis. Examples include finding collaborative players or players with a high sense of social responsibility.

Perceived fairness is a common cause of dispute among players. The richness of players with more playtime on a server is sometimes considered unfair by newcomers. Stealing from other players and destroying their property are common results of trying to counteract that imbalance. This phenomenon is not necessarily a bad thing. Players might enjoy their role as justice fighters and having to protect property adds interesting aspects to the game. It also highlights possibilities to improve collaboration.

New players might feel connected to other newcomers. DiviningRod can help players new to the servers to find each other.

6.5. Indicate Contributions

Our analysis of contributions to common goods indicate correlations with collaboration. We assume players prefer collaboration with altruists to collaboration with egoists. But the game provides no easy way to determine the extent of a player's egoism. Making data available to other players on how much a player gives and takes from shared resources could increase collaboration. Players could find altruist players more easily and reduce the risk of falling for freeloaders. Players might also be interested in improving their own score, which will benefit the server's community.

Contribution scores and the collaboration index for individual players could be published on a server's companion website. Alternatively, a tool similar to Classify could be used to indicate the scores in-game. DiviningRod could be used to find social or antisocial players.

7 Conclusion

We introduce the collaboration index, an universal indicator of collaboration based analysis of different aspects of collaboration in shared virtual worlds. The collaboration index provides a convenient way of comparing amounts of collaboration between different players or servers. This convenient way to quantify collaboration can benefit researchers and administrators trying their virtual worlds in the future.

Our data analysis revealed several possible predictors of collaboration. Players are more likely to collaborate with players they already collaborated with. Players increase their amount of collaboration not by collaborating with the same player more often, but by collaborating longer and with more different players. We found that players who spend a lot of time fighting collaborate more and players who prefer mining collaborate less. We also found collaborative players to be more socially responsible by giving more to shared resources than they take.

We analyzed which pairs of players tend to collaborate. Players prefer to team up with players showing similar interests in respect to building, mining, fighting and exploring. Players with a strong preference for fighting or building are especially likely to collaborate with other fighters or builders respectively. Players doing much fighting but little building tend to team up with their opposites and players with a high specialization on exploring tend to team up with more balanced players. Differences in experience levels have no effect on the likelihood of teaming up.

In order to get those results, we collected over 3452 hours of gameplay from 43 different Minecraft servers. We continue collecting data and expect to gain access to data from even more servers in the future. Our datasets on shared virtual worlds will be a valuable asset for new studies in social science.

We propose several actionable ways to improve collaboration in shared online worlds. Publishing a history of previous server activity or announcing special events can help getting more players into a virtual world at once. Fighters tend to be very collaborative and increasing a server's difficulty and providing supplementary tools can attract fighters. Tools to indicate contributions of individual players or help players find other players can also lead to increased collaboration.

Implementing these insights, we developed DiviningRod, which provides players with programmable social compasses. It can be used to help players find other players based on heuristics found to improve collaboration.

7.1. Limitations

Despite our already-large datasets, our data collection is still at an early stage, and we have left substantive analysis of server-level "cultural" differences for future work. We are still unable to make significant prediction about collaboration on a server-by-server level. A larger dataset would also allow us to determine the significance of smaller correlations.

Our player behavior classifier does not distinguish between fighting monsters and fighting other players. The important role of fighting plays in predicting collaboration suggests that a deeper look into fighting behavior might lead to interesting results.

Because Epilog infers game state only from the stream of game events (and not, say, from the Minecraft world file), it cannot identify buildings, items, or any world state that was introduced before the plugin was installed. However, administrators occasionally reset their world files to allow a player community to "start from scratch." We can record a complete account of a world's development for those there were reset after Epilog was installed.

At present, our suggestions for improving collaboration are based on correlational insights. Field experiments based on DiviningRod and other `HeapCraft` tools will allow us to improve the rigor of our prescriptions. The possibility of controlling the tool remotely and using real-time data from our data collection backend make DiviningRod not only suitable to evaluate many of our proposed ways of improving collaboration, but also for conducting a wide variety of other social experiments.

7.2. Future work

If adoption of `HeapCraft` continues at only the present, modest, rate, we will have data on 250 servers in the next twelve months. At that point, we will have sufficient data to characterize differences in prosociality, collaboration, and community at the server scale. And with a full account of each server's development from the moment of installation, we will be able to characterize, at a micro scale, the social dynamics that distinguish cooperative from uncooperative server cultures. These results will not only enrich the study of collaboration online, they can also provide a much closer look into the emergence of analogous social-scale dynamics as observed in real-world social systems.

From there, the next frontier for virtual world research, and for all large-scale social research, is the controlled experiment. Social scale experiments with thousands to tens of millions of participants are already enabling scientific insights to reach unprecedented scales. Facebook's prominent experiments are currently the state of the art in this domain [KGH14, BFJ$^+$12], though the Music Lab of Salganik and Watts continues to be influential [SW09]. The software platform of Keegan et al. [KOFW$^+$14] was designed specifically for running large-scale collaboration and cooperation experiments. Implementing support for randomized assignment at the server level will add mechanistic underpinning to our currently observational findings, and allow us to directly intervene to develop social systems that are ever better able to foster creativity and collaboration.

A

Data Handling

A high activity server sends millions of events to the logging server each day. In order to offer live data analytics to server administrators, a solution was needed to efficiently process the logged data.

The following analysis is based on data collected from a single server[1] over a period of 7 days. It consists of 12’921’114 logged events.

A.1. Data Transmission

The events are encoded as JSON objects and are sent to the logging server every 20 seconds as a HTTP POST request, one event per line. The request is encrypted with TLS/SSL (https) and the payload is compressed with the deflate algorithm (RFC 1951). The HTTP request header contains a unique token identifying the Minecraft server.

Using JSON provides great flexibility for adding new event properties. Being human readable facilitates debugging and data introspection. It is also widely supported among different programming languages. Since our data contains a lot of duplicate strings (keys, event names, player ids, material names, . . .) it compresses very well. The compression ratio was constantly around 10, which makes the communication protocol comparable to one based on binary data.

Buffering the data for 20 seconds before sending helps keeping the compression ratio high. Furthermore, it reduces communication overhead by reducing the number of HTTP requests. Fewer requests will also reduce the load for the logging server. However, buffering the data will add a delay to the data analytics presented to the server administrators and available to other

[1]Server 4

components like DiviningRod.

A.2. Data Storage

Instead of storing the received data in a database, we use a simple Pile-of-Files scheme. We use a new file per server and day which allows us to use existing file management tools and file system browsers to select data subsets or do backups. The filename contains the date and the server ID in a human readable format.

The files are created by concatenating the unmodified data received by the individual servers. Using different files for different servers allows parallel writing and new content is simply appended to existing files. Using multiple files also keeps the individual file sizes manageable. The low complexity involved in storing recorded data saves computing resources on the logging server and allow the system to be scaled to thousands of servers without problems. The simplicity of the format guarantees the data being usable in the far future even if JSON loses popularity.

Storing the raw data in files solved several problems we noticed from previous attempts of storing the received data directly in a SQL database. Attempts to find common data fields across different types of events and store some events in different tables in order to make the dataset more searchable introduced a lot of complexity. Databases sometimes got corrupted and became inaccessible during development. The huge number of rows made the process of selecting newly added events or accessing events in order of insertion sometimes last seconds. Using text files, accessing events in order of occurrence requires no sorting and new data can be accessed directly by saving the size of a file and continue reading from that position. The file's modification date reveals whether new data has been added without opening the file or executing a query.

In some cases, querying specific data and using pre existing SQL based tools is more convenient than walking thru a text file line by line. For these situations, we create SQL databases optimized for the particular use case by processing the individual files.

A.3. Data Processing

We want some of our data analysis to happen in real-time in order to show up to date data in the Epilog Dashboard and use information about current player behavior for our DiviningRod plugin. We achieved this by creating a system using preemptive multitasking to schedule the processing of data as it arrives.

We create processing classes producing different kinds of aggregated outputs by processing individual events. Those processing classes are able to save their current state, save intermediate results and resume processing additional data afterwards. This allows the scheduler to interrupt the processing in case it takes too long or provide new data as it becomes available. We use 4 different classes which are specialized to produce data for different applications. Using multiple classes allows us to modify individual classes and recalculate their results without affecting other parts of the running system.

For each server, we create dedicated jobs for each of the 4 processing classes. The jobs are a persistent representation of which events already have been processed and manage state and result files. The scheduler uses 2 different priorities, one for real-time (priority 0) and one for background processing (priority 1). Round robin is used among jobs with the same priority.

The scheduler ignores all jobs that are up to date and have reached the end of the event log. It first dispatches all jobs with priority 0. If a job is not finished after 0.2 seconds, it gets interrupted and assigned a lower priority. If all jobs with priority 0 are handled, jobs of priority 1 get executed for up to 1 second since the dispatcher started handling jobs of priority 0. This system allows new data being processed within 1 second after it is received, while allowing newly created jobs to catch up by using the remaining computing time.

A.4. Performance Considerations

We decided to use a high level scripting language for data processing to simplify development and reduce development time. Choosing PHP allows us to reuse a lot of code between the data processing backend and the Epilog Dashboard web frontend. Code written to provide data analysis upon requesting a website can easily be refactored to be used for other kinds of data processing and vice versa.

When processing log files, a lot of time is spent reading data from disk and parsing JSON. These tasks are implemented natively in PHP and using a compiled language yields no significant benefit. Reading millions of lines from a text file and locating a substring took less than two times more time in PHP than by using a program written in C.

Our previous attempt to produce up to date using a SQL database required around one second to compute simple player statistics and around 10 seconds to create an updated version of a player position heat map. Since the results had to be recalculated every 20 seconds when we received new data from a server, it didn't scale well to multiple servers.

The new system produces intermediary datasets which are updated by only adding new data, which usually takes no longer than a couple of milliseconds. An example of such a dataset could be a JSON object containing the total walk distance and duration of gameplay for individual players, or a matrix containing how long players are staying in certain areas. Player statistics are then calculated from the intermediary datasets directly when requested. Other data requires additional processing. Position heat map images for example can take up to a couple hundreds of milliseconds, depending on the method used to create the bitmap image from the corresponding matrix. The bitmaps are only created when requested, and cached until new data is available.

B

User Manuals

B.1. DiviningRod Manual

DiviningRod gives you programmable compasses that can point to any location, object, or player. Predefined compasses help you find players with specific characteristics. New targets can be created by placing signs. Use this plugin to tell others about interesting places or to find people to play with.

The compasses are represented by divining rods which are represented by glowing sticks that wiggle. Every divining rod has a tag which tells it where to point to. If a tag has multiple targets, the most relevant will be used. While holding a divining, the distance to the target will be shown in the item name. Walk around to cause the rod to wiggle and update the distance. Create new targets by writing hashtags on signs.

B.1.1. Usage

A divining rod is created by converting a wooden stick. Use the command **/find <tag>** while holding a stick item in game. A new tag can be assigned by reusing the command. **/find stick** reverts the divining rod to a regular stick. If the player has no item in hand, a stick will be given automatically. Multiple rods with different tags can be created, stored in chests and given to other players. Use different sticks to avoid retyping commands. A right click will show the distance in the console, in case the chat is covering the distance. This also shows the number of targets which can be found with the current tag.

If you don't want to be found, use the command **/dr hide on** to stop any divining rod from eading to you. **/dr hide off** reverses the process and **/dr hide** shows the current setting.

B.1.2. Available tags

- **spawn** points to spawn.
- **player** points to the nearest players.
- **op** points to the nearest operator.
- **@<user>** points to the player with the given name.
- **#<tag>** points to the nearest sign containing the given hashtag.

If Epilog is installed, the following tags are available also:

- **pro** points to the server's most experienced player.
- **socializer** points to the server's most social players.
- **explorer** points to the player having traveled the furthest.
- **miner** points to the player having broken the most blocks.
- **builder** points to the player having placed the most blocks.
- **loner** points to the least social player.
- **settler** points to the player having traveled the least.
- **noob** points to the server's newest player.
- **friend** points to a player to be friends with. (Currently alias for *socializer*)

More tags will be introduced in future plugin versions. New tags based on data analysis can be added in the future by the Epilog logging server if the Epilog plugin is installed. Use the whitelist configuration to avoid previously unknown tags.

B.1.3. Advanced

Some features are disabled by default to prevent unexpected behavior. Enable and disable them with the commands **/dr advanced on** and **/dr advanced off**.

If a target points to more than one target, they are ordered by distance or relevance. The list can be reversed by prepending the tag with a minus sign. -player will for example lead you to the furthest player.

Advanced mode allows you to cycle thru the list of targets. A right click while holding shift will ignore the current top target and show you the next best match. When using the player tag, second closest player will be targeted, then the third and so forth. All targets are reenabled when no more targets are left.

Using the -player tag will never let you reach a target if there are more than 3 players in the same world. A different player will be targeted as soon as you are to close to the previous target. To solve this problem, a target can be locked with a left click while holding shift. This also allows to keep looking for a sign location after the sign has been destroyed.

The color of the item name turns yellow if there are ignored targets and green if the target is locked. This will help you figuring out why the divining rod doesn't show the expected target. All modifications are removed when right clicking in lock mode (left click, right click while holding shift).

The modifications will apply to all rods with the same tag. The modifications are saved between logins, but will be reset if the server restarts or when deactivating advanced mode. To avoid accidentally breaking blocks, no blocks can be broken while holding shift in advanced mode.

The distance to the target ignores the height difference by default. Use the command **/dr 3d on** to get the actual distance.

/find is an alias of **/dr** and the two commands are interchangeable.

B.1.4. More on hashtag signs

The hashtags don't actually point to the sign, but the block the sign is attached to. Multiple tags can be used by either putting several hashtags in one sign or placing additional signs on a block. The targets are removed if the sign or the block breaks. If the block is destroyed by fire, the removal might be delayed.

Hashtags created while the plugin is inactive are not registered. To update and see all registered hashtags for a block, right click the block or an attached sign while holding a divining rod.

B.2. Graph Miner Manual

Graph Miner works best with Chrome. Use Enter to update the Graph after changing options.

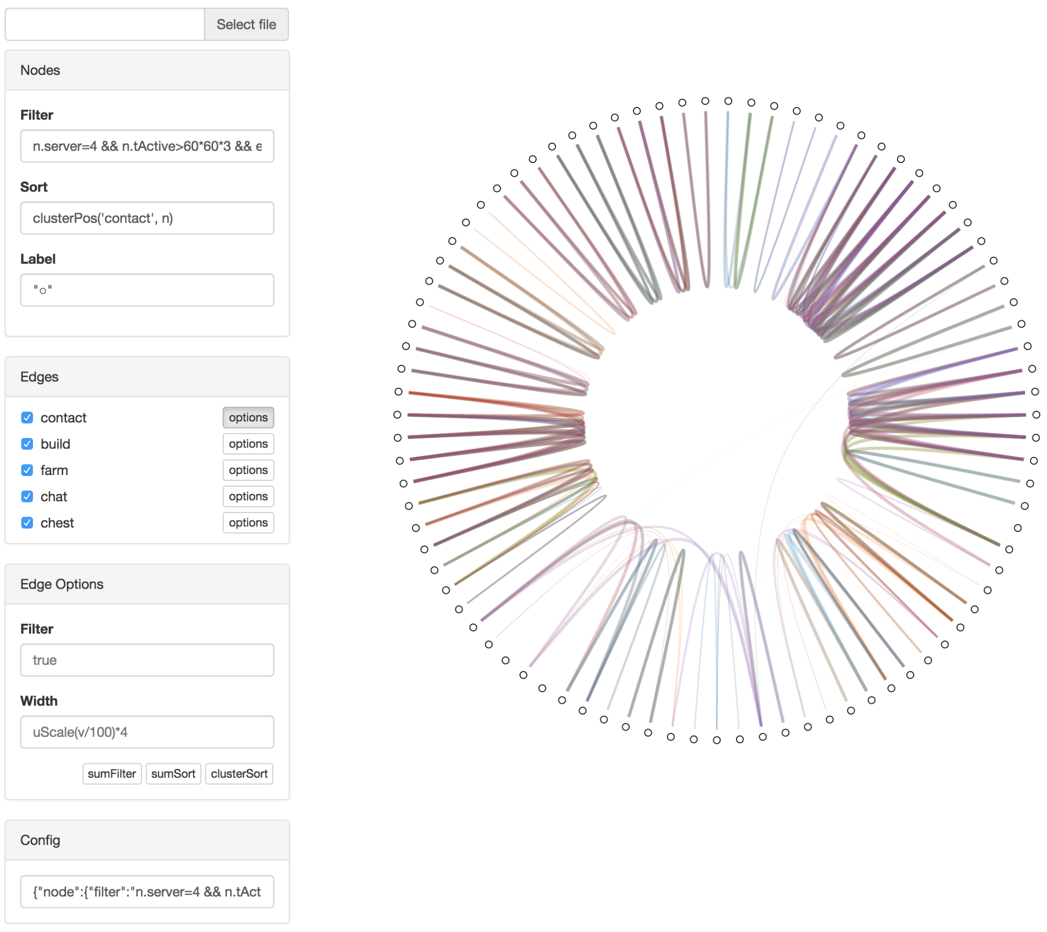
B.2.1. Graph Definition

Graph Miner supports analyzing multiple graphs in parallel, as long as the graphs use the same set of nodes. The input needs to be a JSON object (called **graph** from here on). The nodes are defined as **graph.node.id = id: id**. Additional properties can be defined at will. Edge groups are defined as **graph.edge.groupName = g**. They are represented as sparse weighted adjacency matrix: **g[idSrc][idDst] = weight**.

B.2.2. Player Node Properties

The following properties are available for player nodes created by the HeapCraft framework:

- **n.id** node id
- **n.uid** Mojang account uuid [String]
- **n.name** player name [String]



Nodes

Filter
n.server=4 && n.tActive>60*60*3 && e

Sort
clusterPos('contact', n)

Label
"○"

Edges

- ☑ contact
- ☑ build
- ☑ farm
- ☑ chat
- ☑ chest

Edge Options

Filter
true

Width
uScale(v/100)*4

Config

```
{"node":{"filter":"n.server=4 && n.tAct
```

Figure B.1.: *The Graph Miner tool with the control panel (left) and the resulting visualization (right).*

- **n.tLastEvent** time last recorded event [epoch]
- **n.tActive** active gameplay [seconds]
- **n.tSocial** active near other players [seconds*nPlayers]
- **n.nBlockBreak** number of blocks broken [blocks]
- **n.nBlockPlace** number of blocks placed [blocks]
- **n.sMove** distance moved [blocks]

B.2.3. Expressions

The fields Filter, Sort, Label and Width accept JavaScript expressions. Depending on context, a node object **n** or a value **v** can be accessed.

In addition to standard JavaScript functions, some helper functions are available:

- **edgeSum(Node n)** returns the sums of the weights from all outgoing edges. Can be used for node sorting and filtering.
- **uScale(v)** returns $1 - (1/(v + 1))$ which can be used to normalize edge widths.
- **clusterPos(String edgeGroup, Node n)** returns a node's position after sorting a hierarchical cluster consisting of the edges from edgeGroup. Can be used for node sorting.

B.2.4. Node Options

Filter

Expression to decide if a node is displayed (true) or not (false). The current node object can be accessed over the variable **n**.

Example To show all players active for more than one hour, enter **n.tActive > 60*60**.

Sort

JavaScript expression returning either a string or a number which is used to sort the nodes. The current node object can be accessed over the variable **n**.

Example To sort the nodes by name, enter **n.name**.

Label

JavaScript expression returning the content for the node label. The current node object can be accessed over the variable n.

Example To show player names, enter n.name.

Edge Options

Edge groups can be hidden using the checkboxes in the edge group list. To access a edge group options, use the “options” button. Push it again to hide the options.

Filter

Expression to decide if a edge is displayed (true) or not (false). The weight of the current edge can be accessed over the variable **v**.

Example To only show edges with weights over 9000, enter **v > 9000**.

Width

Expression returning the with of the line representing the edge. The weight of the current edge can be accessed over the variable **v**.

Example To use a logarithmic scale, enter **Math.log(v/100)*2**.

Helpers

sumFilter puts **edgeSum(’edgeGroup’, n) != 0** into the node filter field, resulting in hiding all nodes with no outgoing edges. **sumSort** puts **-edgeSum(’edgeGroup’, n)** into the node sort field, sorting the nodes by the sum of their outgoing edges. **clusterSort** puts **clusterPos(’edgeGroup’, n)** into the node sort field, sorting the nodes by hierarchical clustering.

Config

JSON string containing the current options. Can be copy/pasted and saved elsewhere.

C

Supplementary Lists and Figures

C.1. Epilog Events

The following lists contains the number of times a particular event has been recorded, ordered by frequency.

PlayerMoveEvent (33689409), **PlayerInteractEvent** (1074547), **InventoryContent** (613698), **BlockDamageEvent** (583672), **BlockBreakEvent** (421672), **PlayerToggleSprintEvent** (420754), **PlayerItemHeldEvent** (413704), **PlayerPickupItemEvent** (404385), **PlayerItemInHandEvent** (380765), **EntityDamageByPlayerEvent** (294526), **InventoryCloseEvent** (257360), **PlayerRegisterChannelEvent** (171173), **EntityPortalEnterEvent** (162010), **BlockPlaceEvent** (144799), **InventoryOpenEvent** (131333), **PlayerTeleportEvent** (107662), **PlayerVelocityEvent** (101236), **FoodLevelChangeEvent** (89835), **PlayerInteractEntityEvent** (84790), **PlayerDamageByEntityEvent** (80462), **PlayerRegainHealthEvent** (76383), **PlayerToggleSneakEvent** (69401), **PlayerExpChangeEvent** (67603), **ProjectileLaunchEvent** (45428), **PlayerDamageEvent** (39976), **PlayerDropItemEvent** (37516), **PlayerCommandPreprocessEvent** (30039), **AsyncPlayerChatEvent** (26184), **ChestContent** (23574), **PlayerToggleFlightEvent** (19382), **FakeBlockDamageEvent** (19216), **FakeBlockBreakEvent** (19201), **PlayerDamageByPlayerEvent** (18993), **PlayerItemConsumeEvent** (18655), **CraftItemEvent** (17946), **PlayerLevelChangeEvent** (17086), **PlayerChangedWorldEvent** (16174), **ArmorContent** (13698), **PlayerDeathEvent** (8800), **PlayerLoginEvent** (7650), **EnderChestContent** (5771), **EntityDamageByBlockEvent** (4400), **PlayerRespawnEvent** (4387), **EntityCombustByEntityEvent** (3330), **EntityShootBowEvent** (3309), **PlayerJoinEvent** (3238), **FurnaceExtractEvent** (2788), **PlayerBucketEmptyEvent** (2362), **PlayerBucketFillEvent** (2182), **PlayerQuitEvent** (2173), **EntityCombustByBlockEvent** (1528), **PlayerGameModeChangeEvent** (1517), **PlayerDamageByBlockEvent** (1442), **EntityPortalExitEvent** (1361), **PlayerItemBreakEvent** (1270), **PlayerPortalEvent** (1147),

EntityCombustEvent (406), **BlockIgniteEvent** (314), **PlayerFishEvent** (265), **PlayerKickEvent** (254), **PlayerShearEntityEvent** (200), **HangingPlaceEvent** (169), **PlayerBedLeaveEvent** (152), **PlayerBedEnterEvent** (149), **SignChangeEvent** (127), **PlayerChatTabCompleteEvent** (85), **PlayerLeashEntityEvent** (61), **StructureGrowEvent** (44), **PlayerEggThrowEvent** (27), **PlayerUnleashEntityEvent** (22), **PlayerEditBookEvent** (10)

C.2. Block Types

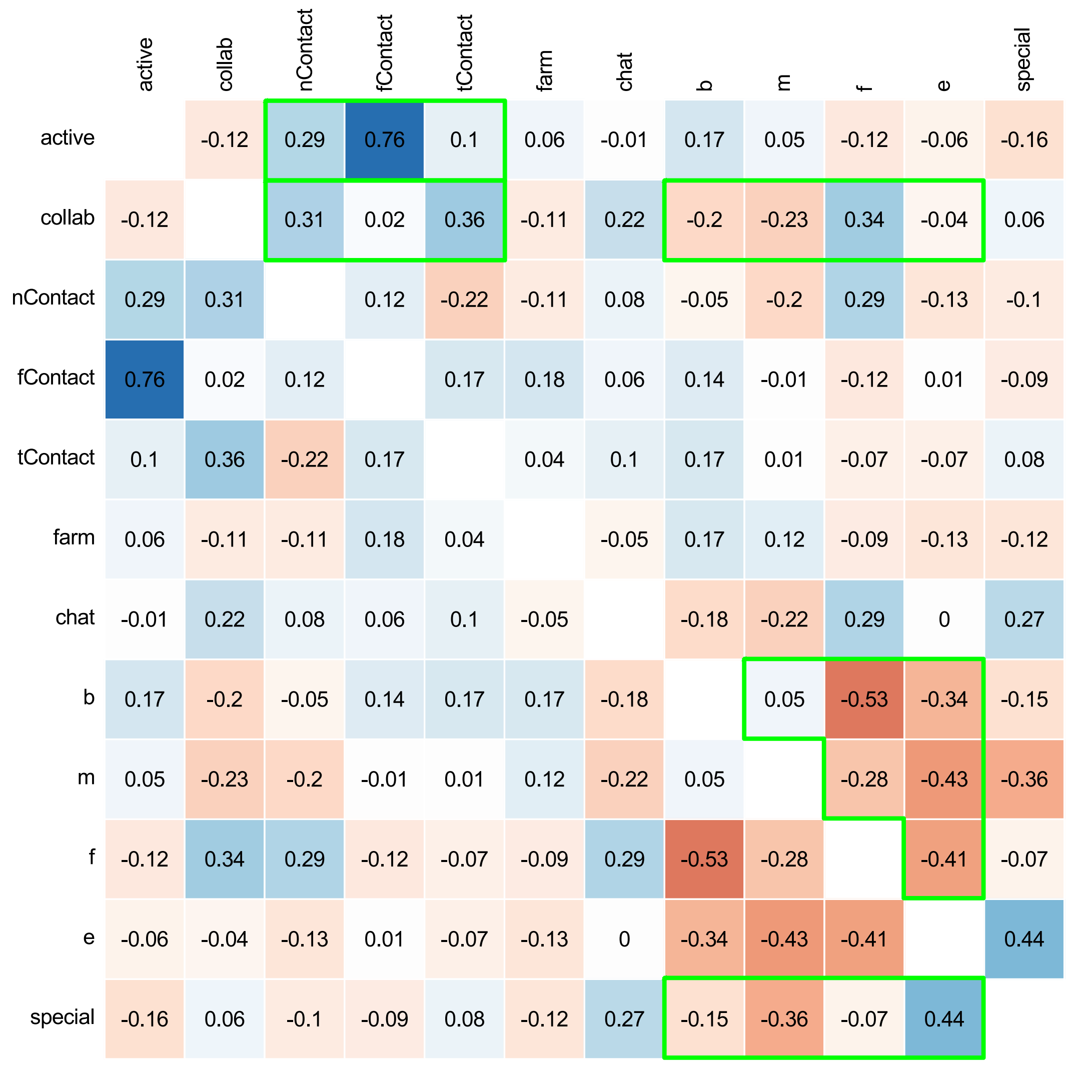
C.2.1. Building Blocks

Blocks of the following materials are used to detect building: ACACIA_DOOR, ACACIA_FENCE, ACACIA_FENCE_GATE, ACACIA_STAIRS, BED_BLOCK, BIRCH_DOOR, BIRCH_FENCE, BIRCH_FENCE_GATE, BIRCH_WOOD_STAIRS, BOOKSHELF, BRICK, BRICK_STAIRS, CARPET, CLAY, COAL_BLOCK, COAL_ORE, COBBLESTONE, COBBLESTONE_STAIRS, COBBLE_WALL, DARK_OAK_DOOR, DARK_OAK_FENCE, DARK_OAK_FENCE_GATE, DARK_OAK_STAIRS, DIAMOND_BLOCK, DIAMOND_ORE, DIRT, DISPENSER, DOUBLE_STEP, DOUBLE_STONE_SLAB2, EMERALD_BLOCK, EMERALD_ORE, ENDER_STONE, FENCE, FENCE_GATE, GLASS, GLOWING_REDSTONE_ORE, GLOWSTONE, GOLD_BLOCK, GOLD_ORE, GOLD_PLATE, GRAVEL, HARD_CLAY, HAY_BLOCK, ICE, IRON_BLOCK, IRON_DOOR_BLOCK, IRON_FENCE, IRON_ORE, IRON_PLATE, IRON_TRAPDOOR, JACK_O_LANTER JUKEBOX, JUNGLE_DOOR, JUNGLE_FENCE, JUNGLE_FENCE_GATE, JUNGLE_WOOD_STAIRS, LAPIS_BLOCK, LAPIS_ORE, LEAVES, LEAVES_2, LOG, LOG_2, MOSSY_COBBLESTONE, NETHERRACK, NETHER_BRICK, NETHER_BRICK_STAIRS, NETHER_FENCE, NOTE_BLOCK, OBSIDIAN, PACKED_ICE, PISTON_BASE, PISTON_EXTENSION, PISTON_MOVING_PIECE, PISTON_STICKY_BASE, PRISMARINE, QUARTZ_BLOCK, QUARTZ_ORE, QUARTZ_STAIRS, REDSTONE_BLOCK, REDSTONE_LAMP_OFF, REDSTONE_LAMP_ON, REDSTONE_ORE, RED_SANDSTONE, RED_SANDSTONE_STAIRS, SAND, SANDSTONE, SANDSTONE_STAIRS, SLIME_BLOCK, SMOOTH_BRICK, SMOOTH_STAIRS, SNOW, SNOW_BLOCK, SOUL_SAND, SPONGE, SPRUCE_DOOR, SPRUCE_FENCE, SPRUCE_FENCE_GATE, SPRUCE_WOOD_STAIRS, STAINED_CLAY, STAINED_GLASS, STAINED_GLASS_PANE, STEP, STONE, STONE_PLATE, STONE_SLAB2, THIN_GLASS, TNT, TRAP_DOOR, WOOD, WOODEN_DOOR, WOOD_DOUBLE_STEP, WOOD_PLATE, WOOD_STAIRS, WOOD_STEP, WOOL.

C.2.2. Farming Blocks

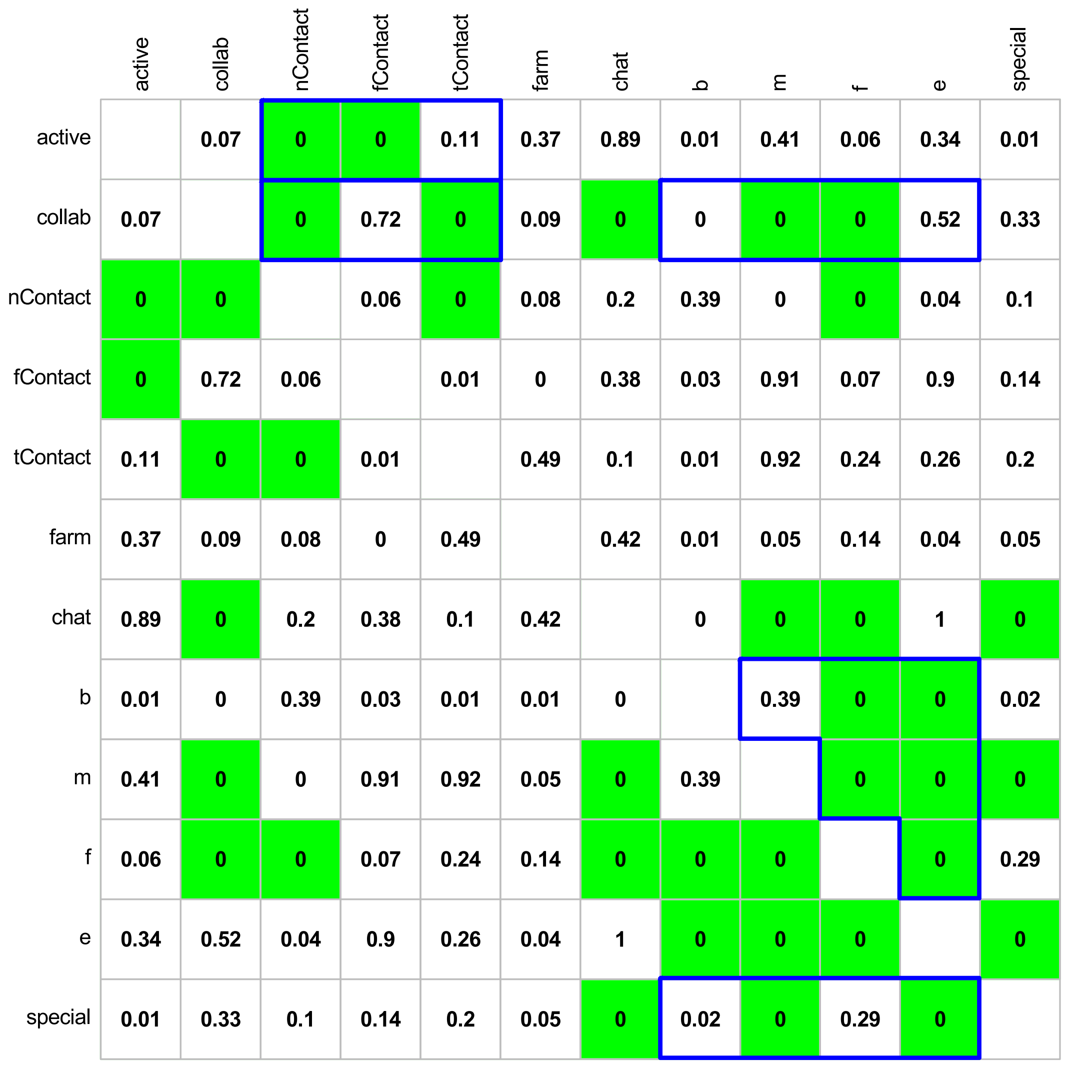
Blocks of the following materials are used to detect farming: BROWN_MUSHROOM, CACTUS, CARROT, CROPS, MELON_BLOCK, MELON_STEM, NETHER_WARTS, POTATO, PUMPKIN, PUMPKIN_STEM, RED_MUSHROOM, SUGAR_CANE_BLOCK.

C.3. Player Correlations



	active	collab	nContact	fContact	tContact	farm	chat	b	m	f	e	special
active		-0.12	0.29	0.76	0.1	0.06	-0.01	0.17	0.05	-0.12	-0.06	-0.16
collab	-0.12		0.31	0.02	0.36	-0.11	0.22	-0.2	-0.23	0.34	-0.04	0.06
nContact	0.29	0.31		0.12	-0.22	-0.11	0.08	-0.05	-0.2	0.29	-0.13	-0.1
fContact	0.76	0.02	0.12		0.17	0.18	0.06	0.14	-0.01	-0.12	0.01	-0.09
tContact	0.1	0.36	-0.22	0.17		0.04	0.1	0.17	0.01	-0.07	-0.07	0.08
farm	0.06	-0.11	-0.11	0.18	0.04		-0.05	0.17	0.12	-0.09	-0.13	-0.12
chat	-0.01	0.22	0.08	0.06	0.1	-0.05		-0.18	-0.22	0.29	0	0.27
b	0.17	-0.2	-0.05	0.14	0.17	0.17	-0.18		0.05	-0.53	-0.34	-0.15
m	0.05	-0.23	-0.2	-0.01	0.01	0.12	-0.22	0.05		-0.28	-0.43	-0.36
f	-0.12	0.34	0.29	-0.12	-0.07	-0.09	0.29	-0.53	-0.28		-0.41	-0.07
e	-0.06	-0.04	-0.13	0.01	-0.07	-0.13	0	-0.34	-0.43	-0.41		0.44
special	-0.16	0.06	-0.1	-0.09	0.08	-0.12	0.27	-0.15	-0.36	-0.07	0.44	

Figure C.1.: *Correlations between player variables. Specific groups of analyses of interest are bounded in green.*



	active	collab	nContact	fContact	tContact	farm	chat	b	m	f	e	special
active		0.07	0	0	0.11	0.37	0.89	0.01	0.41	0.06	0.34	0.01
collab	0.07		0	0.72	0	0.09	0	0	0	0	0.52	0.33
nContact	0	0		0.06	0	0.08	0.2	0.39	0	0	0.04	0.1
fContact	0	0.72	0.06		0.01	0	0.38	0.03	0.91	0.07	0.9	0.14
tContact	0.11	0	0	0.01		0.49	0.1	0.01	0.92	0.24	0.26	0.2
farm	0.37	0.09	0.08	0	0.49		0.42	0.01	0.05	0.14	0.04	0.05
chat	0.89	0	0.2	0.38	0.1	0.42		0	0	0	1	0
b	0.01	0	0.39	0.03	0.01	0.01	0		0.39	0	0	0.02
m	0.41	0	0	0.91	0.92	0.05	0	0.39		0	0	0
f	0.06	0	0	0.07	0.24	0.14	0	0	0		0	0.29
e	0.34	0.52	0.04	0.9	0.26	0.04	1	0	0	0		0
special	0.01	0.33	0.1	0.14	0.2	0.05	0	0.02	0	0.29	0	

Figure C.2.: *p-values for correlations between player variables. Significant correlations with p<0.01 are highlighted green. Specific groups of analyses of interest are bounded in blue.*

C.4. Combined Graphs

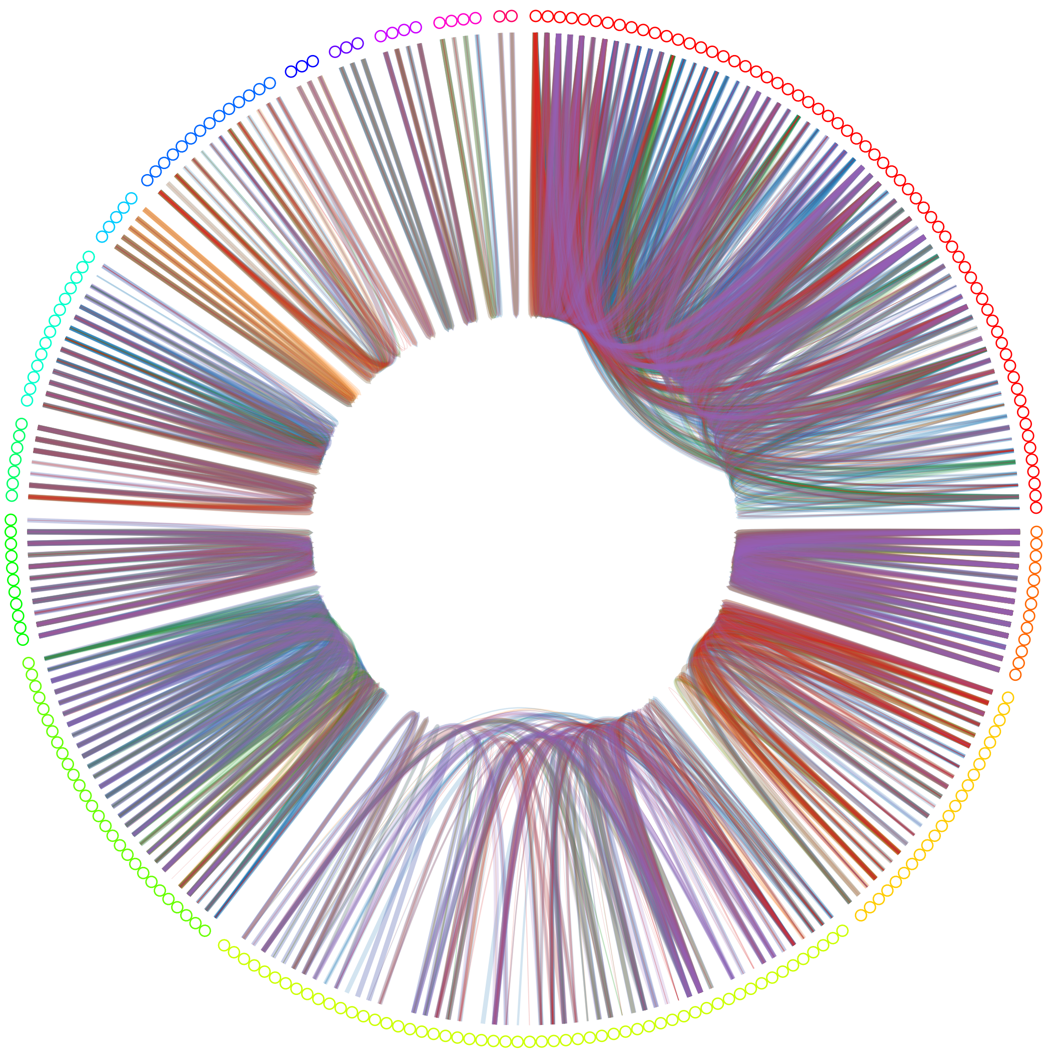


Figure C.3.: *Edges of all 5 graphs combined for all servers. contact (blue), build (orange), farm (green), chat (red), chest (purple)*

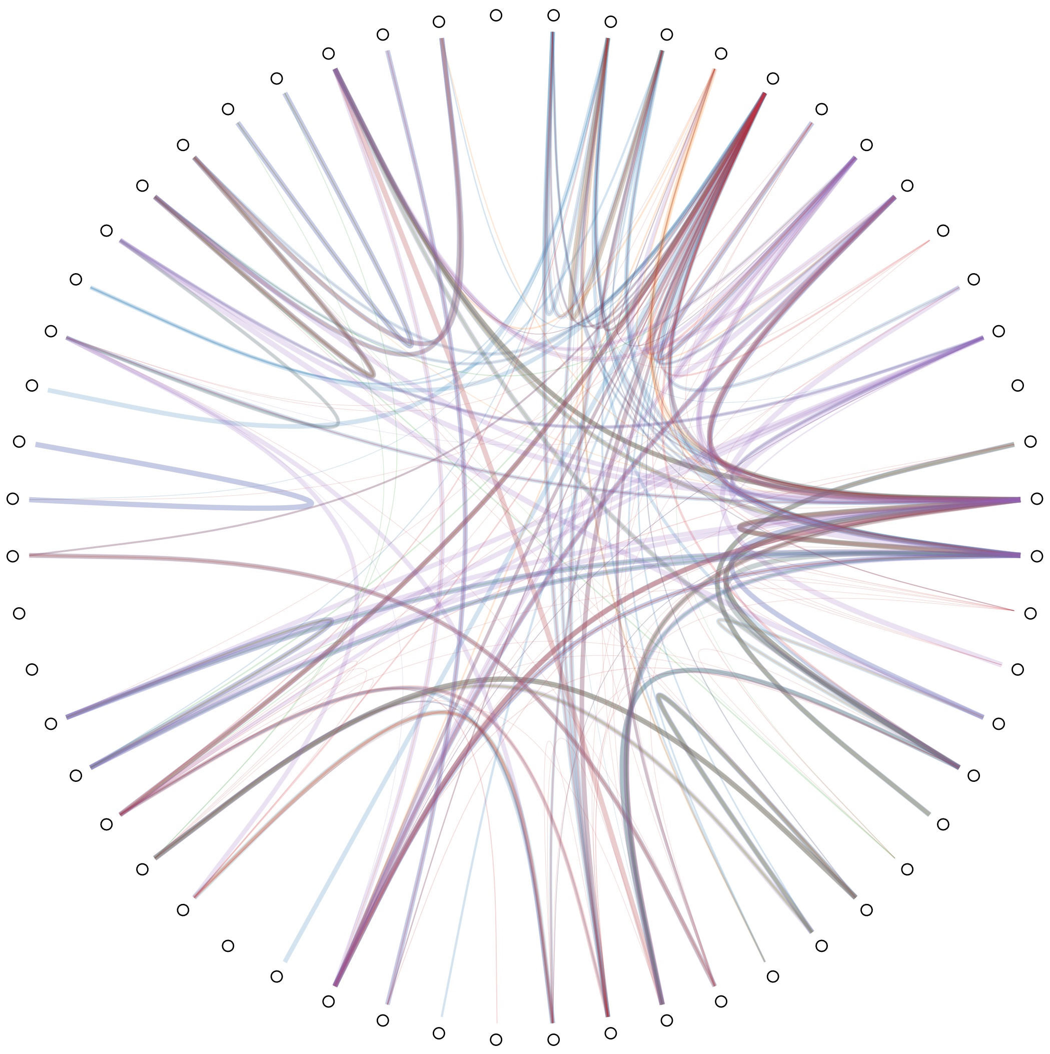


Figure C.4.: Edges of all 5 graphs combined for server 4. contact (blue), build (orange), farm (green), chat (red), chest (purple)

Bibliography

[Bai07] William Sims Bainbridge. The scientific research potential of virtual worlds. *science*, 317(5837):472–476, 2007.

[Bav50] A Bavelas. Communication patterns in task-oriented groups. *Journal of the acoustical society of america*, 22(6):725–730, 1950.

[BCW$^+$14] I I Bukvic, C Cahoon, A Wyatt, T Cowden, and K Dredger. OPERAcraft: Blurring the Lines between Real and Virtual. *Proceedings ICMC*, 2014.

[BFJ$^+$12] Robert M Bond, Christopher J Fariss, Jason J Jones, Adam D I Kramer, Cameron Marlow, Jaime E Settle, and James H Fowler. A 61-million-person experiment in social influence and political mobilization. *Nature*, 489(7415):295–298, September 2012.

[Cas06] E Castronova. On the Research Value of Large Games: Natural Experiments in Norrath and Camelot. *Games and Culture*, 1(2):163–186, April 2006.

[Cas08] Edward Castronova. A test of the law of demand in a virtual world: exploring the Petri dish approach to social science. *econstor.eu*, 2008.

[CHC$^+$14] Taejoong Chung, Jinyoung Han, Daejin Choi, Taekyoung Ted Kwon, Huy Kang Kim, and Yanghee Choi. Unveiling group characteristics in online social games. In *the 23rd international conference*, pages 889–900, New York, New York, USA, 2014. ACM Press.

[Che09] Mark Chen. Communication, Coordination, and Camaraderie in World of Warcraft. *Games and Culture*, 4(1):47–73, 2009.

[CRK13] Edward Castronova, Travis L Ross, and Isaac Knowles. Designer, Analyst, Tinker: How Game Analytics Will Contribute to Science. In M Seif El-Nasr and al, editors, *Game Analytics: Maximizing the value of player data*, pages 665–687. Springer-Verlag, London, March 2013.

[CWS$^+$09] E Castronova, D Williams, Cuihua Shen, R Ratan, Li Xiong, Yun Huang, and B Keegan. As real as real? Macroeconomic behavior in a large-scale virtual world. *New Media & Society*, 11(5):685–707, July 2009.

[DY13] Nicolas Ducheneaut and Nick Yee. Data collection in massively multiplayer online games: Methods, analytic obstacles, and case studies. In *Game Analytics*, pages 641–664. Springer, 2013.

[DYNM07] Nicolas Ducheneaut, Nicholas Yee, Eric Nickell, and Robert J Moore. The life and death of online gaming communities. In *the SIGCHI conference*, pages 839–848, New York, New York, USA, 2007. ACM Press.

[FSN$^+$14] David J French, Brett Stone, Thomas T Nysetvold, Ammon Hepworth, and W Edward Red. Collaborative Design Principles From Minecraft With Applications to Multi-User CAD. In *ASME 2014 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, pages V01BT02A039–V01BT02A039. ASME, August 2014.

[Gar14] Nate Garrelts, editor. *Understanding Minecraft: Essays on Play, Community and Possibilities*. McFarland, 2014.

[GS55] H Guetzkow and H Simon. The impact of certain communication nets upon organization and performance in task-oriented groups. *Management Science*, 1(3/4):233–250, 1955.

[HM51] G Heise and G Miller. Problem solving by small groups using various communication nets. *Journal of Abnormal and Social Psychology*, 46(3):327–335, 1951.

[HSC13] Yun Huang, Cuihua Shen, and Noshir S Contractor. Distance matters: Exploring proximity and homophily in virtual world networks. *Decision Support Systems*, 55(4):969–977, November 2013.

[HYBC13] Yun Huang, Wenyue Ye, Nicholas Bennett, and Noshir Contractor. Functional or social? In *the 2013 conference*, pages 399–408, New York, New York, USA, 2013. ACM Press.

[KAW$^+$10] Brian Keegan, Muhammad Aurangzeb Ahmed, Dmitri Williams, Jaideep Srivastava, and Noshir Contractor. Dark Gold: Statistical Properties of Clandestine Networks in Massively Multiplayer Online Games. In *2010 IEEE Second International Conference on Social Computing (SocialCom)*, pages 201–208. IEEE, August 2010.

[KGH14] A D I Kramer, J E Guillory, and J T Hancock. Experimental evidence of massive-scale emotional contagion through social networks. *Proceedings of the National Academy of Sciences of the United States of America*, June 2014.

[KOFW$^+$14] Brian Keegan, Katherine Ognyanova, Brooke Foucault Welles, Christoph Riedl, Ceyhun Karbeyaz, Waleed Meleis, David Lazer, Jason Radford, and Jefferson Hoye. Conducting Massively Open Online Social Experiments with Volunteer

Science. *Second AAAI Conference on Human Computation and Crowdsourcing*, 2014.

[Lea13] Alex Leavitt. Crafting Minecraft: Negotiating Creative Produsage-Driven Participation in an Evolving Cultural Artifact. *Selected Papers of Internet Research*, 3:1–32, 2013.

[MFS$^+$15] O Mryglod, B Fuchs, M Szell, Yu Holovatch, and S Thurner. Interevent time distributions of human multi-level activity in a virtual world. *Physica A*, 419(1):681–690, 2015.

[MKF$^+$ar] Müller, Kapadia, Frey, Klingler, Mann, Solenthaler, Sumner, and Gross. Statistical analysis of player behavior in minecraft. In *Proceedings of the 10th International Conference on Foundations of Digital Games*, FDG '15, To Appear.

[Oko09] Chitu Okoli. A Brief Review of Studies of Wikipedia in Peer-Reviewed Journals. In *International Conference on Digital Society (ICDS)*, pages 155–160. IEEE, 2009.

[Red] The average minecraft player, 4800 responses in 7 hours. enjoy these charts and graphs! `https://www.reddit.com/r/Minecraft/comments/lncf7/the_average_minecraft_player_4800_responses_in_7/`. Accessed: 2015-05-20.

[SC13] Catherine Schifter and Maria Cipollone. Minecraft as a teaching tool: One case study. In *Society for Information Technology & Teacher Education International Conference*, 2013.

[SENDC13] Magy Seif El-Nasr, Anders Drachen, and Alessandro Canossa, editors. *Game Analytics*. Springer London, London, 2013.

[SLT10] Michael Szell, Renaud Lambiotte, and Stefan Thurner. Multirelational organization of large-scale social networks in an online world. *PNAS*, 107(31):13636–13641, August 2010.

[SW09] Matthew J Salganik and Duncan J Watts. Web-Based Experiments for the Study of Collective Social Dynamics in Cultural Markets. *Topics in Cognitive Science*, 1(3):439–468, July 2009.

[Wag04] C Wagner. Wiki: A technology for conversational knowledge management and group collaboration. *The Communications of the Association for Information ...*, 2004.