筑波大学大学院博士課程

システム情報工学研究科特定課題研究報告書

写真共有サービスのユーザ動向可視化・分析システムの開発
―データベースの設計・実装およびアクティブユーザ予測モデルの実装―

河越 嵩介

修士（工学）

（コンピュータサイエンス専攻）

指導教員　田中 二郎

2016年3月

概要

我々のプロジェクトが取り扱う RoomClip はインテリアに特化した SNS である。RoomClip に登録することでユーザは自分の好きな部屋やインテリアの写真を投稿することができ、さらに投稿した写真上で他のユーザとコミュニケーションを取ることができる。このため RoomClip ではユーザのアクセス数を増加させて投稿写真数を増加させることがサービスの維持、ひいては発展に繋がる。しかし現状、RoomClip のユーザのアクセス率（1 週間に 1 回以上アクセスしたことのあるユーザの割合）は、5%〜10%にとどまっている。そこで Tunnel 社からサービス活性化のために「ユーザの行動の変化を詳細に把握し、施策がユーザの行動に与えている影響を知りたい」との要望があった。要望を実現するために、我々はユーザの動向の分析及び可視化を行うシステム—— ARC (Analysis for the RoomClip) ——の開発を行った。

筆者は本プロジェクトにおいて、ARC のデータベースの設計・実装を担った。また、非アクティブになるだろうと予測されるユーザには適切な施策を打つ必要があるため、非アクティブユーザを予測するモデルの実装を行い、73%の精度での予測を実現した。さらに、ユーザの中から特徴的なユーザ群を割り出すためにユーザアクティビティの因子分析を行い、全体の半分の割合の中にいくつかの特徴をもったユーザ群を探り出した。

目 次

図 目 次

表 目 次

第1章　序論

1.1　インテリア写真共有サービス RoomClip

本プロジェクトの分析対象である RoomClip[1] はインテリアに特化した無料の SNS である。部屋やインテリアを好む人が、自分の部屋やインテリアを撮影して投稿したり、他のユーザが投稿した写真を見たりして楽しむことができる。RoomClip には男女を問わず様々な写真が投稿されている。ログインしたユーザは投稿された写真に対して、「いいね」という好評価をつけたり、自分だけのお気に入りに登録したり、コメントを投稿したりすることで、様々なコミュニケーションをとっている。「いいね」が多かったり、お気に入りの登録が多かったり、コメントの数が多い写真は「話題のインテリア」、「オススメのインテリア」としてトップページで紹介される。また写真を投稿する際には「IKEA」や「一人暮らし」などのその写真の性質を表すタグを自由につけることができ、これが RoomClip 上での様々なコミュニティの形成に役立っている。RoomClip に投稿された写真は誰でも自由に閲覧することができるが、特定のユーザをフォローすることでそのユーザの行動を逐一知ることができる。大勢のフォロワーがいたり、投稿した写真に数多くの「いいね」をもらったりしているユーザは「話題のユーザ」としてトップページで紹介される。また、投稿写真の中に気に入ったインテリアがあれば、サービス上で自由に購入することもできる。

RoomClip は Tunnel 株式会社が 2012 年初頭より運営している。現在ユーザ数は 30 万人を越えており、投稿写真は 100 万枚を越えており、月間 PV は 5000 万を越えている [2]。企業とのタイアップを活発的に行ったり、ユーザが投稿した写真からインテリアのムック本を出版したりすることで収益を上げている。

1.2　問題意識

RoomClip のメインコンテンツは投稿写真なので、ユーザのアクセス数を増加させて、投稿写真数を増加させることがサービスの維持、ひいては発展に繋がる。アクセス数を増加させるためには新規顧客を獲得することも考えられるが、既存顧客を維持することも重要な要素である。何故なら、新規顧客をいくら獲得しても彼らに既存顧客として定住してもらえなければ、サービスは衰退してしまうからである。既存顧客の維持として、「カレンダーの置き方」や「こたつのある部屋」などのコンセプトを定めてユーザの投稿を募り、その中から Tunnel 社によって選出された写真を投稿したユーザに対して賞を与える、などのコンテスト形式のイベントが実施されている。

現在の RoomClip におけるユーザのログイン率（1 週間に 1 回以上ログインしたユーザの割合）は 5%〜10%にとどまっている。そこで Tunnel 社はユーザのログイン率を優先して上げるべく、ユーザの行動の変化を詳細に把握し、施策がユーザの行動に与えている影響を知る必要があると考えている。

1.3 本報告書の構成

第 2 章では本プロジェクトの概要について述べる。第 3 章では本プロジェクトで行った分析手法について述べる。第 4 章では開発した分析システム ARC について述べる。第 5 章ではデータベースの設計・実装について述べる。第 6 章では非アクティブユーザを予測するモデルの実装について述べる。第 7 章ではユーザアクティビティの因子分析の実装について述べる。第 8 章ではシステムを開発するにあたって参考にした既存の分析ツールについて述べる。第 9 章では開発したシステムの評価について述べる。第 10 章ではプロジェクト全体を振り返っての所感について述べる。最後に、第 11 章で結論を述べる。

本プロジェクトにおいて筆者が担当した部分は第 5 章、第 6 章、第 7 章である。

第2章　プロジェクトの概要

本章では、プロジェクトの体制や目的、スケジュールについて述べる。

2.1　プロジェクトの体制

本プロジェクトの顧客は Tunnel 株式会社である。課題担当教員として、岡瑞起准教授および橋本康弘助教からご助力を頂く。開発は古谷、崔、万、および河越の四人で開発を行う。以下に本プロジェクトの体制表を示す。

<table>
<tr><td colspan="2">顧客</td><td>Tunnel 株式会社</td></tr>
<tr><td rowspan="6">筑波大学</td><td rowspan="2">課題担当教員</td><td>岡 瑞起</td></tr>
<tr><td>橋本 康弘</td></tr>
<tr><td rowspan="4">学生</td><td>河越 嵩介</td></tr>
<tr><td>古谷 翔太</td></tr>
<tr><td>崔 野</td></tr>
<tr><td>万 シャンシャン</td></tr>
</table>

表 2.1: 本プロジェクトの体制

2.2　顧客の要望

顧客からはサービス活性化のために、「ユーザの行動の変化を詳細に把握し、施策がユーザの行動に与えている影響を知りたい」との要望があった。

2.3　プロジェクトの目的とアプローチ

そこで本プロジェクトでは、「Tunnel 社が RoomClip のユーザの動向を把握できるシステムの開発」を目的とする。アプローチとして、ユーザを 7 種類の行動に基づいて分類し、経時変化を定量的・定性的に可視化・予測する。

第3章　分析について

本章では、分析対象とする RoomClip 上でユーザが行ったログデータの概要およびその分析方法を具体例を挙げながら述べる。

3.1　分析対象とするデータ

ユーザの動向を分析するにあたって、Tunnel 社から次の 7 つのアクションに関するログデータを頂いた。これらのアクションはいずれも RoomClip 上でユーザが行えるアクションである。それぞれについて説明する。

アクション	説明
view	ユーザが RoomClip の何れかのページに アクセスしたときのログ
like	あるユーザが他のユーザの投稿した写真に対して、 「いいね！」ボタンを押したときのログ
clip	あるユーザが他のユーザの投稿した写真に対して、 「お気に入り」のボタンを押したときのログ
follow	あるユーザが他のユーザをフォローしたときのログ
comment	あるユーザが他のユーザの投稿した写真に対して コメントをしたときのログ
post	ユーザが写真を投稿したときのログ
favorite-tag	ユーザがあるタグをお気に入りにしたときのログ。 iOS 及び Android アプリで実装されており、 Web 版では実装されていない.

表 3.1: 7 つのアクション

3.2　ユーザの分類

まず我々は、コホート分析 [3] を参考にし、上述した 7 つのアクションをそれぞれ行ったことがあるかないかの 2 値でユーザを分類することにした。2 値分類にした理由としては、行うことへの敷居が高いアクション (e.g., Comment、Post) があると考え、行ったことがあるかな

いかの 2 値で分類することで比較的大きな意味を持つクラスタに分類することができる、というのが 1 つ。もう 1 つは、行ったことがあるかないかの 2 値で分類することで、あるアクションを行っていないユーザ群に対して適切な施策を打つことができるから、という理由である。

3.2.1 分類の数

分類の数は、view を行っていることを前提とした上で 6 つのアクションをそれぞれ行ったことがあるかないかの 2 値による分類が 64 通りあり、また view を行っていないのが 1 通りあるので、合わせて 65 通りになる。この 65 個の分類先を本プロジェクトではノードと呼ぶ。この 65 個のノードの 2 つ（もしくはそれ以上）の期間の結果を比較することでユーザの動向を時系列的に分析することができる。

3.2.2 分類対象とする期間

加えて述べなければならないことは、2 値分類の対象とする期間の日数についてである。例えば、1 日ごとにユーザを分類すると、おそらく多くのユーザがほとんど何のアクションも行っていない、または view しか行っていないという結果が返ってくるだろう。しかし 7 日ごとに集計して分類を行うとすれば、1 日ごとの場合よりある程度バラつきが抑えられる結果になるだろう。一方で 30 日ごとに集計して分類すれば、さらにバラつきが抑えられるだろうが、逆にバラつきが無さすぎてほとんどのユーザが全てのアクションを行っているグループに分類されてしまうだろう。この例からわかるように、この分類の対象とする日数の最適解は一意には定まらない。我々としては 1 週間、つまり 7 日がある程度ユーザを分類するのに都合がいいと考えているが、この日数に関しては Web アプリケーション上で自由に変更できるように実装している。

以上をまとめると、我々は、ユーザを、ある日数の中で、7 つのアクションをそれぞれ行ったことがあるかどうかの 2 値分類を行って 65 コのノードのいずれかに分類し、その結果を異なる期間で比較する。

3.2.3 ユーザの分類の例

次に分類の例を示す。例えば、あるユーザの 1 月 1 日〜1 月 7 日までのアクティビティの集計結果が view20 回、comment10 回、post 1 回で、他のアクションを 1 回も行っていないとする。このユーザをそれぞれのアクションを行ったか行っていないかの 2 値で分類すると、このユーザは view あり、like なし、clip なし、follow なし、comment あり、post あり、favtag なしというノードに分類される。このように、一定期間の間に行ったことがあるかないかの二値化を全てのユーザに対して行い、各ユーザを 65 コのノードのいずれかに分類する。

アクション	回数
view	20
like	0
clip	0
follow	0
comment	10
post	1
favtag	0

表 3.2: あるユーザの 1 月 1 日〜1 月 7 日のアクティビティ

アクション	あり／なし
view	あり
like	なし
clip	なし
follow	なし
comment	あり
post	あり
favtag	なし

表 3.3: あるユーザの 1 月 1 日〜1 月 7 日におけるノード

第4章　分析システム ARC

本プロジェクトで我々は分析システム——ARC（Analytics for RoomClip）——を開発した。ARC は Web アプリケーションであり、Web ブラウザ上で動作する。ARC には 5 つの機能——3 つの画面と 2 つの API があり、本章ではそれらについて述べる。

4.1　システムの概要

ARC は Ranking、Flow、Dashboard とユーザの動向を分析するための 3 つの画面を有する。以下の表にそれぞれの大まかな役割を示す。

画面名	役割
Ranking	ノードに所属するユーザの変動人数の表示
Flow	ノード間のユーザの流出入の表示
Dashboard	7 つのアクションの利用率の表示

表 4.1: 各画面の役割

それぞれの画面について、具体例を挙げながら述べる。

4.1.1　Ranking

この画面は、3.2 で述べた分析方法に基づいて、ノードに所属するユーザの変動人数の観点から可視化した画面である。65 コのノードを期間 A から期間 B への変動人数・変動率の値の大小の順に並べ替えて表示する。この画面で、期間内で最も動きのあったノードのユーザを確認することができる。画面の例を次に示す。

この画面は、実施した施策の効果を検証するために使うことを想定している。例えば、Tunnel 社がユーザのアクティブ率を上げるべく、clip を促進する施策を実施したとする。施策を実施する前の先週と、実施した後の今週との違いを見るため、分類の対象とする日数を 7 日に設定し、先週の日付と今週の日付を入力して画面を更新する。続いて何のアクションを起こしたユーザが増えたのかを確認するため、変動人数でソートする。この結果、clip を行ったユーザが増えていることが確認できた。

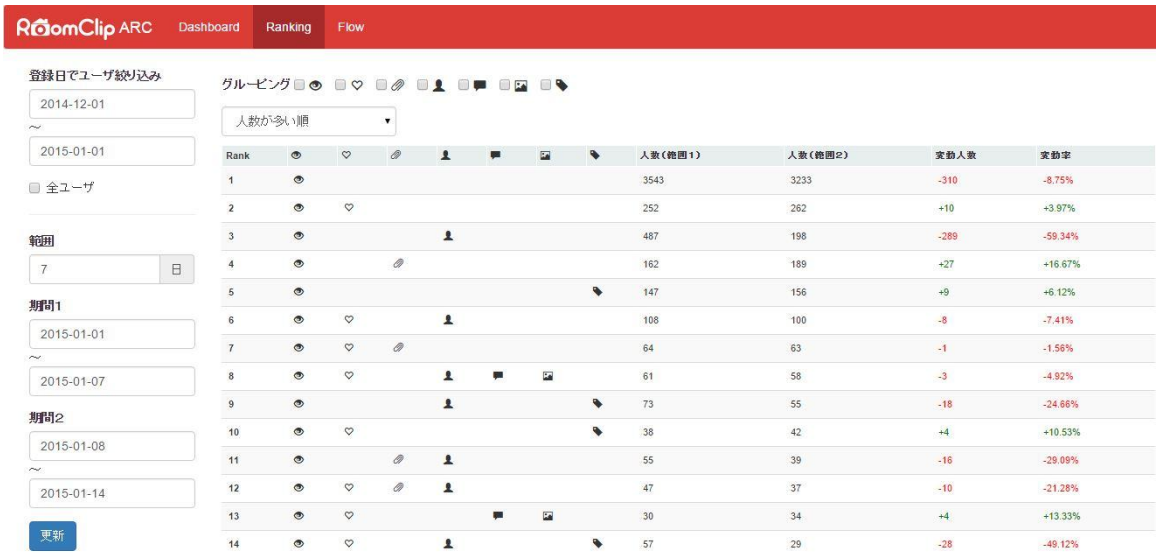


図 4.1: Ranking 画面の例

4.1.2 Flow

この画面は、3.2 で述べた分析方法に基づいて、ノード間のユーザの流出入の観点から可視化した画面である。手法として Sankey Diagram[4] を用い、ある期間 A からある期間 B への、65 コのノードに属するユーザの流出入を示す。この画面で、期間 A での特定のユーザが期間 B でどのような行動を行ったか、逆に期間 B での特定のユーザが期間 A でどのような行動を行っていたか、というユーザの行動を時系列で分析できる。画面の例を次に示す。

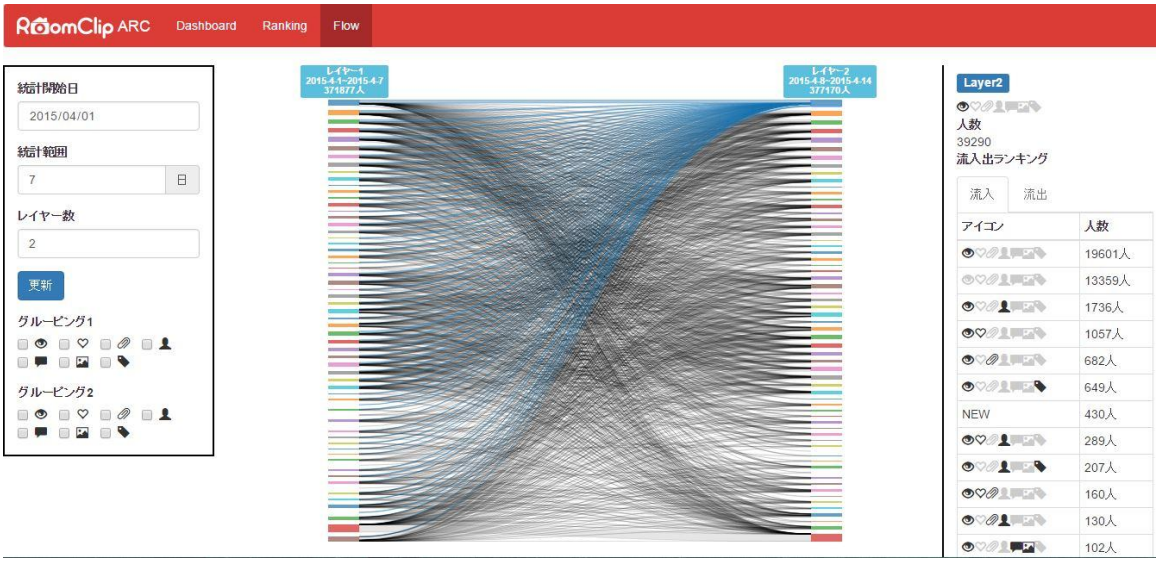


図 4.2: Flow 画面の例

この画面は、施策を打つための判断材料として使うことを想定している。例えば、Tunnel 社はサービスから離れたユーザを再びサービスに戻すための施策を行いたい、と考えているとする。この場合は、ある週で何のアクションも行わなくなったユーザが翌週にどのアクションを行ったノードへ一番多く移動しているかを見ることで調べることができる。

4.1.3 Dashboard

この画面は、ユーザのコホートの可視化とは別に、7 つのアクションのそれぞれの利用率を可視化した画面である。7 つのアクションそれぞれについて、その日までにサービスに登録し

ているユーザの中でその日にそのアクションを行ったユーザの割合を折れ線グラフで示している。画面の例を次に示す。

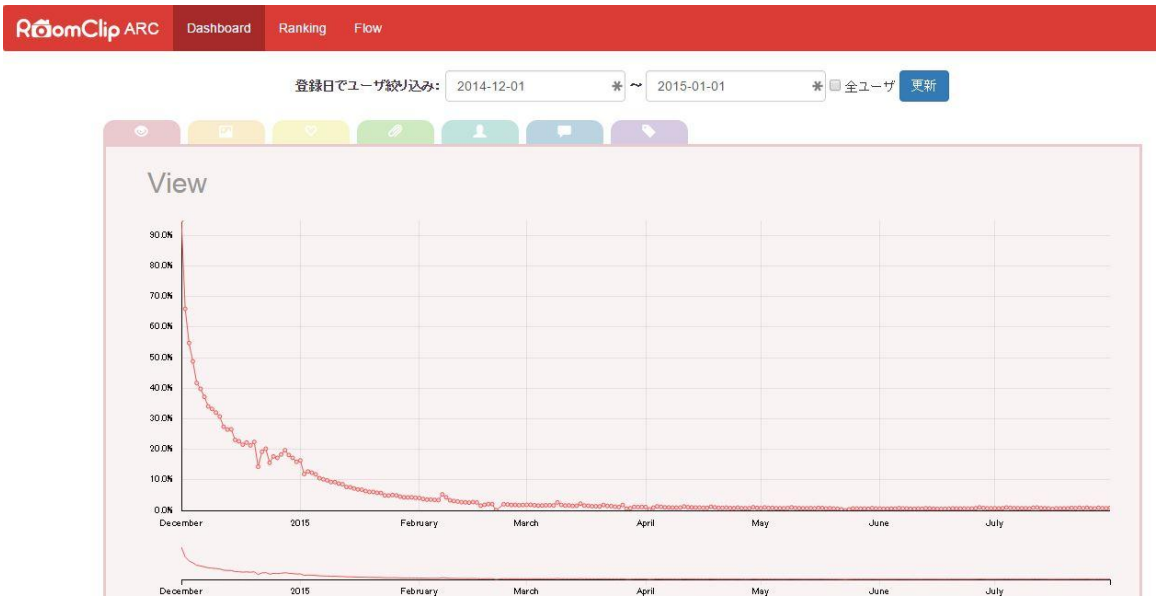


図 4.3: Dashboard 画面の例

この画面は、実施した施策の効果を検証するために使うことを想定している。施策実施後に登録したユーザとそうでないユーザのアクションの利用率を比較し、施策がどれくらい影響を与えたかを判断する。

4.1.4 2 つの API

ARC には先述した 3 つの画面の他に、2 つの API がある。1 つは非アクティブユーザを予測する API であり、もう 1 つはユーザアクティビティの因子分析を行う API である。これらは筆者が担当した部分であるため、詳細については後述する。

4.2 システムの構成

本システム ARC の構成図を以下に示す。

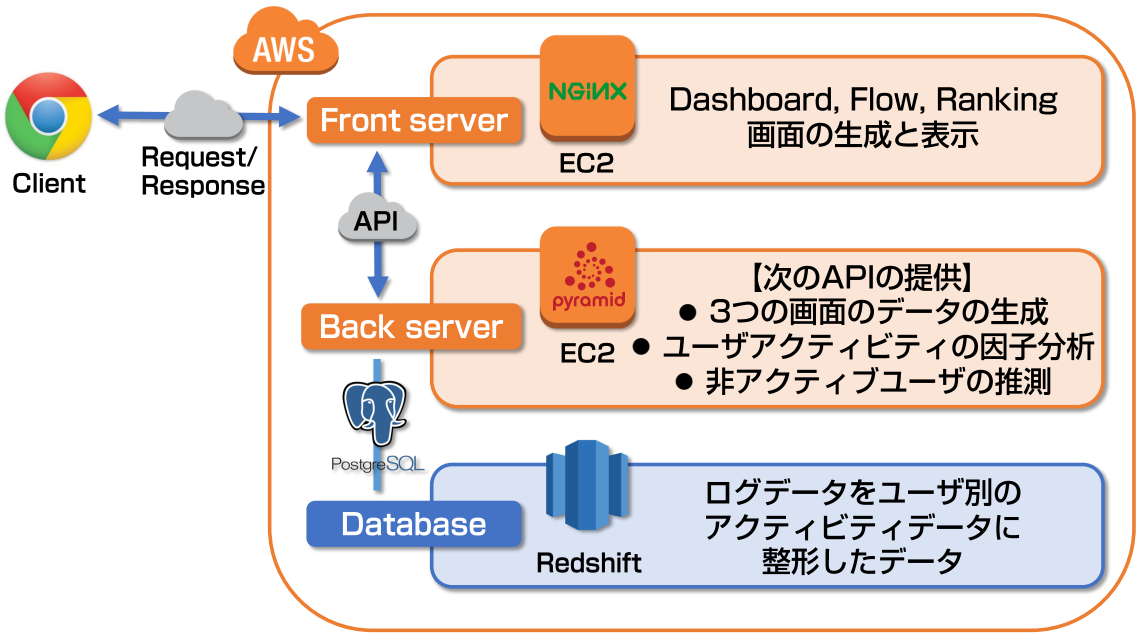


図 4.4: システムの構成図

ARC はアーキテクチャとしてマイクロサービス [5] を採用している。マイクロサービスとは、一連の小さなサービスで 1 つのアプリケーションを開発する手法である。本プロジェクトではサーバを、「フロントサーバ」と「バックサーバ」の 2 台で構築することで、開発速度の向上及びコストの削減を図っている。具体的には、フロントサーバでは Flow 画面、Ranking 画面の生成と表示を行っており、バックサーバではデータベースから取得したデータを、フロントサーバで表示する Ranking データと Flow データにそれぞれ整形する処理を API として実装している。

また ARC では、クラウドサービスとして Amazon Web Service（以降 AWS と略す）を利用しており、2 台のサーバには EC2 を、データベースには Redshift を用いている。AWS を選んだ理由としては、まずクラウドサービスとして豊富なサービスを揃えており、また登録開始から 1 年間は基本的なサービスを無料で使うことができ、さらに我々の顧客である Tunnel 社が既に AWS を利用しており、インスタンスのコピーを納品することで Tunnel 社の ARC の導入コストを下げることができると考えたためである。

4.3 開発体制

ARC の開発体制について述べる。

4.3.1 担当範囲

以下に開発におけるメンバーの担当範囲を示す。

メンバー	担当範囲
河越	データベースの設計・実装およびアクティブユーザ予測モデルの実装
古谷	サーバサイドの設計・実装
崔	データ可視化と UI の設計と実装
万	フロントエンドの UI デザインの設計

表 4.2: 担当範囲

チームのリーダーは古谷であり、彼がチームのマネジメントを行った。

4.3.2 開発スケジュール

本プロジェクトの開発スケジュールの実績を次に示す。

図 4.5: 開発スケジュールの実績

本プロジェクトでは一度に一年分のスケジュールを立てていない。7 月、10 月、11 月に行われた Tunnel 社との打ち合わせを一つのマイルストーンとし、常に次のマイルストーンまでの計画を立てて開発を行った。このように開発を行った理由としては、プロジェクト発足当初は Tunnel 社自身も本当に必要なものが明確になっていなかったので、必要最小限の機能から実装していくことで、本当に必要な機能だけを実装するためである。

要件定義フェーズでは、個人作業というよりはチーム全体で開発の参考となる分析ツールの調査や RoomClip の現状の調査を行って、要件を定義していった。7 月上旬に 1 回目の Tunnel 社との打ち合わせを行い、そこで要件が明確に定まった。打ち合わせ後に、定義された要件に沿って開発する分析システムの像を固めていき、システムの設計と担当の振り分けを行った。

ここから第一次開発フェーズに入った。このフェーズでは、チーム全体でのまとまった行動から個人個人の行動に移っていった。筆者はデータベースの設計・実装の担当となったので、まずデータベースの選定を行った。データベースの選定が済み次第、サーバサイドの担当で

ある古谷と協力して必要となるテーブルを洗い出していった。その後データベースの実装に入った。データベースの実装は 9 月半ばで終え、その後は機械学習の調査に取り掛かった。

10 月上旬に 2 回目の Tunnel 社との打ち合わせを行い、ARC のフィードバックを頂いた。そのフィードバックを元に、改良点をいくつか洗い出し、それらに優先度をつけて実装することとなった。ここから第二次開発フェーズに入り、筆者は非アクティブユーザを予測するモデルの実装を開始した。

11 月下旬に 3 回目の Tunnel 社との打ち合わせを行い、その後 ARC を 1 週間ほど使っていただいた。この期間が試用期間にあたる。試用期間を終え、Tunnel 社から頂いたフィードバックを元に更なる改良を行うこととなった。ここから第三次開発フェーズに入り、筆者はユーザアクティビティの因子分析のモジュールを実装した。また開発フェーズでは必要に応じて Tunnel 社とメールにて連絡を取った。

そして 2 月上旬に納品を行う予定である。

第5章　ARCのデータベース

本章では、筆者の担当範囲であるARCのデータベースの設計・実装およびデータベースとバックサーバ間の連携について述べる。

5.1　データベースの選定

ARCでAWSを利用することは既に決まっていたので、データベースはそこから選ぶことになった。AWSでは無料で使えるデータベースにRDS、DynamoDB、Redshift[7]（2ヶ月間のみ）の3種類が用意されている。筆者はこの中からRedshiftを用いることにした。理由としては、列指向データベースであるからである。後述するが、ARCデータベースが1日に各ユーザが各アクションをどれだけ行ったかというデータを持つため、列指向データベースであるRedshiftと相性がいいのである。またRedshiftはPostgreSQLをベースに作られているので、操作になじみやすかったり、データのロードもストレージであるS3を経由することで簡単に行うことができたりすることも理由の一つである。しかしRedshiftには1点問題がある。それは先述したように、無料期間が最初の2ヶ月のみであることである。そこで我々は、最初の2ヶ月で必要となる主なテーブルを構築及びロードすることで、最初の2ヶ月以降は1年間無料で使えるRDSで代用し、必要がある場合のみRedshiftを用いることとした。RDSはAWSが提供するRDBMSサービスであり、MySQL、Oracle、PostgreSQLを提供している。当然Redshiftに比べると性能は1段落ちるが、開発環境としては十分に使える性能である。以下に、開発に用いたRedshiftとRDSのバージョンを示す。

データベース	バージョン
Redshift	dc1.large
RDS	db.t2.micro

表5.1: データベースのバージョン

5.2　ログデータの整形

分析対象としたデータは3.1で述べた、RoomClipのユーザが2014年の10月から翌年の5月までに行った7つのアクションの利用ログデータである。このログデータをARCで利用するために整形を行った。

アクションのログデータには、基本的にそのアクションを行ったユーザとそのアクションが行われた日時が格納されており、その他にそのアクション特有のカラムが格納されている。そこでまずは 3.2 の分析がしやすいように日の単位でグルーピングを行い、ある日にあるユーザがそのアクションを何回行ったか、という形式にデータを整形した。以下に整形前と整形後のデータの形式を示す。

カラム名	データ型	説明
created	TIMESTAMP	アクションが行われた日時
user_id	INTEGER	アクションを行ったユーザ ID
unique_column		各アクション固有のカラム （view の場合は view したページの種類、 like の場合は like した photo_id、など）

表 5.2: 整形前のデータ

カラム名	データ型	説明
date	DATE	アクションが行われた日
user_id	INTEGER	アクションを行ったユーザ ID
action_count	SMALLINT	アクションの回数

表 5.3: 整形後のデータ

そして整形した各アクションのデータを結合して後述する user_activity のような形にした。この際、view 以外のアクションは同日に view を最低 1 回以上行っているという前提があるため、整形後の view のテーブルにそれ以外のアクションのテーブルを外部結合することで実現した。

Workbench ツールとして、Redshift では Aginity Workbench for Redshift を、RDS では SQL Manager Lite for PostgreSQL を利用した。

5.3 テーブルの定義

Redshift および RDS 上に構築した 3 つのテーブル——user_activity、node、user_info について述べる。

5.3.1 user_activity

このテーブル user_activity は、1 日毎に各ユーザが各アクションをどれだけ行ったかを表すテーブルである。user_activity の定義を以下に示す。

各カラムについて説明していく。

No	カラム名	データ型	Not Null	sort key	dist key
1	u_date	DATE	Yes	1	Yes
2	u_id	INTEGER	Yes	2	
3	u_view	SMALLINT	Yes		
4	u_like	SMALLINT	Yes		
5	u_clip	SMALLINT	Yes		
6	u_follow	SMALLINT	Yes		
7	u_comment	SMALLINT	Yes		
8	u_post	SMALLINT	Yes		
9	u_favtag	SMALLINT	Yes		

表 5.4: user_activity の定義

(1)u_date
日付である。このテーブルが持つデータの期間は 2014-10-08 〜 2015-05-21 である。

(2)n_id
ユーザ ID である。このテーブルが持つユーザ ID はその日に何らかのアクションを 1 回以上行っているユーザ ID だけであり、何のアクションも行っていないユーザ ID は持たない。

(3)u_view、u_like、u_clip、u_follow、u_comment、 u_post、u_favtag
その日そのユーザの行った 7 つのアクションのそれぞれの累計数である。これらのカラムは大きな数値になることもないため、データ型は SMALLINT としている。また前述したように、この user_activity は各アクションのデータを結合したものである。よって view 以外のアクションが行われたときは最低 view が 1 回以上行われているという前提があるので、view が 0 の場合はその他 6 つのアクションのそれぞれの累計数も 0 である。

このテーブルでは sortkey を u_date、u_id の順に指定している。sortkey を用いることで、このキーを指定した順番にデータをソートすることができる。また distkey には u_data を指定している。distkey は一つのカラムのみ指定することができ、このキーで指定されたカラムを基準にデータを分散させることができる。

5.3.2 node

このテーブル node は、ノード ID と 7 つのアクションそれぞれ行ったか行っていないかの 2 値分類の対応付けを表すテーブルである。node の定義を以下に示す。
各カラムについて説明していく。

(1)n_id
3.2.1 で前述した 65 コのノード ID である。ID は 0 から始まり、順に 1 つずつ増えていく。

No	カラム名	データ型	Not Null	sort key	dist key
1	n_id	DATE	Yes	1	Yes
2	n_view	SMALLINT	Yes		
3	n_like	SMALLINT	Yes		
4	n_clip	SMALLINT	Yes		
5	n_follow	SMALLINT	Yes		
6	n_comment	SMALLINT	Yes		
7	n_post	SMALLINT	Yes		
8	n_favtag	SMALLINT	Yes		

表 5.5: node の定義

(2)n_view、n_like、n_clip、n_follow、n_comment、 n_post、n_favtag
ノード ID に対応する 0 か 1 の値がそれぞれに入っている。ただし n_view が 0 のときはそれ以外の全てのカラムも 0 である。

Sortkey には n_id を指定し、テーブルのサイズから distkey には何も指定していない。

5.3.3 usre_info

このテーブル user_info は、ユーザの簡易的な登録情報を示すテーブルである。user_info の定義を以下に示す。

No	カラム名	データ型	Not Null	sort key	dist key
1	created	TIMESTAMP	Yes	1	Yes
2	id	INTEGER	Yes		
3	name	VARCHAR	Yes		

表 5.6: user_info の定義

このテーブルはあるユーザがいつ RoomClip に登録したかを表している。created は TIMESTAMP 型であり、そのユーザが登録された日時を秒数まで表現する。

5.4 テーブルの使われ方

今節で、上述したテーブルが ARC の各機能でどのように使われているかを述べる。

Ranking 画面ではノードに所属するユーザの変動人数の表示のため、user_activity と node の両テーブルが用いられる。手順としては、まず分析の対象とする 2 つ（またはそれ以上）の

期間の 1 日目が指定され、そこから分析対象とする日数の分だけ各ユーザの各アクションの合計が計算される。アクションの合計数が 0 の場合は 0、1 以上の場合は 1 として判断し、その結果を 5.3.2 のテーブルと照合する。そしてある期間に各ユーザがどのノード ID に所属するかを判断し、ノード ID に所属するユーザ数を算出する。

Flow 画面ではノード間の流出入の表示のため、user_activity と node の両テーブルが用いられる。基本的には Ranking 画面の手順と同様であるが、Flow ではノード ID に所属するユーザ数を算出した上で、期間 A の各ノードから期間 B の各ノードへ流出するユーザ数もまた算出している。

Dashbord 画面では 7 つのアクション利用率の表示のため、user_activity と user_info の両テーブルが用いられる。その日の利用率は、その日にアクションを行ったユーザ数をその日までに登録したユーザ数で求められる。user_activity からその日にアクションを行ったユーザ数を算出し、user_info からその日までに登録したユーザ数を算出する。

非アクティブユーザを予測する API では、指定された期間内の user_activity のデータに加え、各ユーザがその週において RoomClip にアクセスした日数から非アクティブユーザの予測を行う。

ユーザアクティビティの因子分析をする API では、user_activity 上の指定された期間内のデータに対して因子分析を行う。

第6章　非アクティブユーザを予測するモデル

本章では筆者が担当した非アクティブユーザを予測するモデルについて述べる。

6.1　意義

既存顧客の維持は企業として非常に重要であり、顧客を維持し続けるために企業はユーザがアクティブでい続けてくれるような施策を実施する。このとき、もし近いうちに非アクティブになるようなユーザとそうでないユーザを予測できるようになるとしたら、それぞれに適した施策を実施することができる。先行研究として、ソーシャルネットワークサービス（以下 SNS と略す）の利用状況を用いて、サービスを近いうちに辞める可能性が高いユーザとそうでないユーザを予測する手法がいくつか提案されている [7][8][9] [10][11][12]。そこで本プロジェクトでもそれらを参考に、ユーザの 1 週間の利用状況のログデータを用いて、翌週のアクティブユーザ・非アクティブユーザを予測するモデルの開発を行った。

6.2　アクティブユーザの定義

本モデルでは、ある N 週間に何らかのアクションを 1 回以上行ったユーザをその週にアクティブなユーザとして定義し、そうでないユーザを非アクティブなユーザとして定義する。定義により、N の値が大きければ大きいほど多くのユーザがアクティブユーザとして見なされ、値が小さければ小さいほど多くのユーザが非アクティブユーザとして見なされる。よってこの N の値を何にするかは非常にデリケートな問題であるが、今回は N = 7 として、モデルを構築した。

6.3　使用データ

アクティブユーザを予測させるにあたって、次の表で示すデータを学習データとした。前述した 5.3.1 の user_activity テーブルを 1 週間で集計した結果に、ユーザの情報や活動日数を追加したものである。データの均衡度合いについては、全 1,447,845 レコード中、アクティブユーザが 962,105、非アクティブユーザが 485,740 とおよそ 2:1 の比率である。以下に使用した学習データを示す。

特徴ベクトル名	説明
view	SMALLINT
like	SMALLINT
clip	SMALLINT
follow	SMALLINT
comment	SMALLINT
post	SMALLINT
favtag	SMALLINT
count_days	SMALLINT

表 6.1: 学習データ

(1)view、like、clip、follow、comment、post、favtag
あるユーザが 1 週間に行った各アクションの累計数である。これらの要素は各アクションごとに正規化を行ってから学習させている。

(2)count_days
ユーザが 1 週間の内にアクションを行った日数である。この値が 5 以上のユーザは高確率で翌週もアクションを行っていることが確認できたため、この要素を採用した。

6.4 予測モデルの実装

分類器としては、Support Vector Machine（以下、SVM と略す）と Random Forests、Xgboost(eXtreme Gradient Boosting)[13] を用いた。SVM は 2 値判別を行う手法であり、アクティブユーザ・非アクティブユーザの予測に効果的だと判断した。カーネルとしては最もよく使われる RBF カーネルを採用した。Random Forests は、説明変数をランダムサンプリングして作成された複数の決定木を用いて目的変数の予測を行う手法である。弱学習機を複数組み合わせるアンサンブル学習により、精度の高いモデルが構築できる。Xgboost は、使い方次第でディープラーニングに匹敵する精度を実現できるため採用した。線形分離不可能なパターンを高い精度で識別する。

API として実装するためにプログラミング言語として python を用い、数値計算環境を構築するために Anaconda を導入した。ライブラリとして scikit-learn、numpy、xgboost を使用した。開発環境として ipython-notebook を利用した。以下にそれぞれのバージョンを示す。

6.5 モデルの評価と考察

6.3 で示したデータを 2:1 の比率で学習データとテストデータとに分割して前述した 3 つのモデルの評価を行った。その結果を以下に示す。

名称	バージョン
python	3.4.3
Anaconda	2.3.0 (64-bit)
scikit-learn	0.16.1
numpy	1.9.2
xgboost	0.4
ipython-notebook	3.2.0

表 6.2: 予測モデルの開発環境のバージョン

	適合率	再現率	F1 スコア
非アクティブユーザ	0.56	0.71	0.63
アクティブユーザ	0.83	0.72	0.77
平均／合計	0.74	0.72	0.72

表 6.3: SVM の評価

	適合率	再現率	F1 スコア
非アクティブユーザ	0.58	0.66	0.62
アクティブユーザ	0.81	0.75	0.78
平均／合計	0.73	0.72	0.73

表 6.4: Random Forests の評価

	適合率	再現率	F1 スコア
非アクティブユーザ	0.57	0.68	0.62
アクティブユーザ	0.82	0.74	0.78
平均／合計	0.74	0.72	0.73

表 6.5: Xgboost の評価

適合率は正（負）と予測したデータのうち、実際に正（負）であるものの割合を表し、再現率は実際に正（負）であるもののうち、正（負）であると予測されたものの割合を表し、F1スコアは適合率と再現率の調和平均を表す。

3つのモデルの比較結果から、モデル毎に大きな差異はないことがみられた。よってモデル単体での考察ではなく、3つのモデルをまとめて考察する。

表から、アクティブユーザクラスの結果は全て7割を越えておりそこそこ評価できるものであるが、非アクティブユーザクラスの結果は芳しくないことがわかる。両者の再現率の差はSVMが0.01、Random Forestsは0.09、Xgboostは0.06とあまり開きはないが、適合率の差はSVMで0.27、Random Forestsで0.23、Xgboostで0.25と20%程の大きな開きが存在する。一方で、アクティブユーザ・非アクティブユーザの平均を取ると再現率より適合率の方が結果が良い。つまり現在の学習データからは、アクティブユーザと予測できるグループと、アクティブユーザとも非アクティブユーザとも予測できないグループとの2つに大別されたことがわかる。よって、これ以上識別結果を向上させるには、アクティブユーザと非アクティブユーザの中から非アクティブユーザを識別する特徴ベクトルが必要ということになる。第3次開発においてそのような特徴ベクトルを探していたが、結局見つけることはできなかった。今回の予測モデルの学習には、アクションの回数のデータを使っているが、「viewしたページの種類やlike・clipした写真の内容」というアクションの質のデータは使っていない。そのようなデータを用いることで、さらにモデルの性能を向上させられるかもしれない。

またRandom ForestsとXgboostで求めた特徴ベクトルの重要度を図化した。それを以下に示す。

図を見ると明らかだが、Random Forestsにおいてはcount_daysが識別するための主な特徴ベクトルとなっていることがわかる。一方、Xgboostではcount_daysの重要度が一番大きいことには変わりないが、それでも4割には満たず他の特徴ベクトルも大きな重要度を誇っている。実際にcount_dyasと翌週のアクティブ・非アクティブの関係を見てみると、count_daysが5以上のユーザ、つまり1週間に5回以上アクセスするユーザは高確率で翌週はアクティブであることが確認でき、1週間に4回以下しかアクセスしないユーザに関しても、アクセスの回数が多いユーザの方が翌週もアクションを行う可能性が高いことが確認できた。このことは石井らのSNSを利用するユーザの1週間における利用頻度の調査結果からも伺える[15]。

Random Forestsではcount_days以外ではviewが3%を占めているだけで、それ以外の要素はほとんど意味がないことが伺える。このことからRandom Forestsは、ユーザのRoomClipのサービスそのものへの関心を予測に用いたと判断できる。

Xgboostではcount_days以外の特徴ベクトルも大きな重要度を占めている。comment、follow、clip、post、favtag、view、likeの順に重要度が大きい。このアクションの順序を見ると、commentやfollowなど他者に関与する行為が大きいことがわかる。同時に、post、clip、likeなどの写真に関与する行為はそこまで大きくないこともわかる。このことからXgboostは、部屋やインテリアの写真そのものよりも、それを通じた他者とのコミュニケーションを予測に用いたと判断できる。

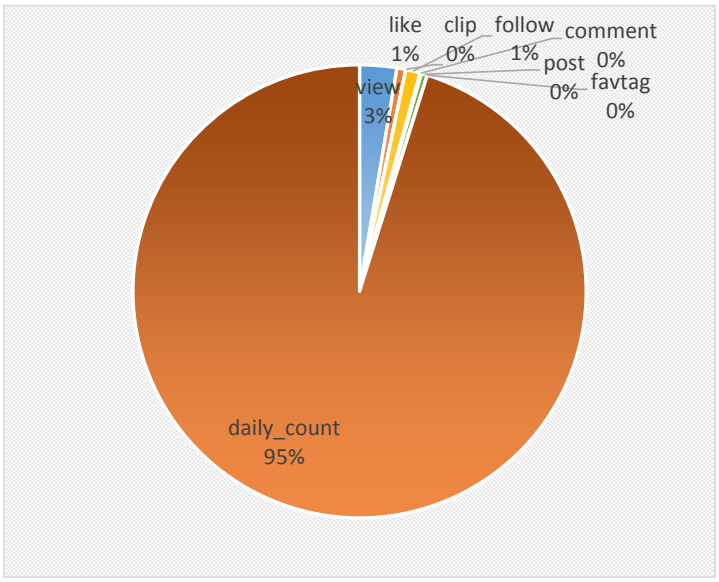


図 6.1: Random Forests の特徴ベクトルの重要度

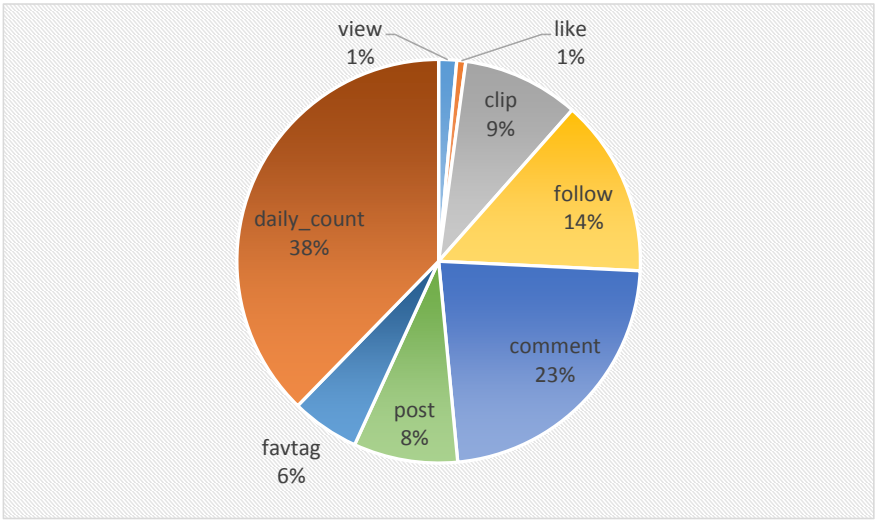


図 6.2: Xgboost の特徴ベクトルの重要度

6.6 API の仕様

最も精度の高かった Xgboost のモデルを API に組み込んで ARC に実装した。この API は、パラメータとして日付（YYYY-MM-DD）を指定することで、その日からの 1 週間に何らかのアクションを起こしたユーザ——つまりアクティブユーザの中から、次の 1 週間で何のアクションも起こさないユーザ——つまり非アクティブユーザを予測し、その予測結果の一覧を返す。例えば 1 月 1 日がパラメータとして指定された場合、1 月 1 日から 1 月 7 日の期間でアクティブだったユーザの中から、1 月 8 日から 1 月 15 日の期間で非アクティブになるだろうと予測されるユーザ ID の一覧を返す。

API 上では次のように動作している。まず指定された日付から 1 週間分のデータを user_activity テーブルから取得する。続いてそのデータを集計して、1 週間に各ユーザが 7 つのアクションをそれぞれ行った回数とユーザが 1 週間の内にアクションを行った日数を算出し、表 6.1 のような形式に変換する。変換後のデータを予測モデルに入力し、その中から非アクティブと予測されたユーザ ID を出力する。

第7章　ユーザアクティビティの因子分析

本章では筆者が担当したユーザアクティビティの因子分析について述べる。

7.1　意義

ARC には 4.1.2 で述べた施策を打つための判断材料をみる、Flow 画面というノード間のユーザの流出入を示す画面がある。この画面では、統計開始日と統計範囲を指定して 65 個のノード間の流出入を示すが、さらに 7 つのアクションでノードをグルーピングすることができる。例えば like と comment でグルーピングをすると次の図のように、like と comment をしたユーザ、like をして comment をしなかったユーザ、like をしなくて comment をしたユーザ、like と comment をしなかったユーザという 4 つの集合にグルーピングできる。

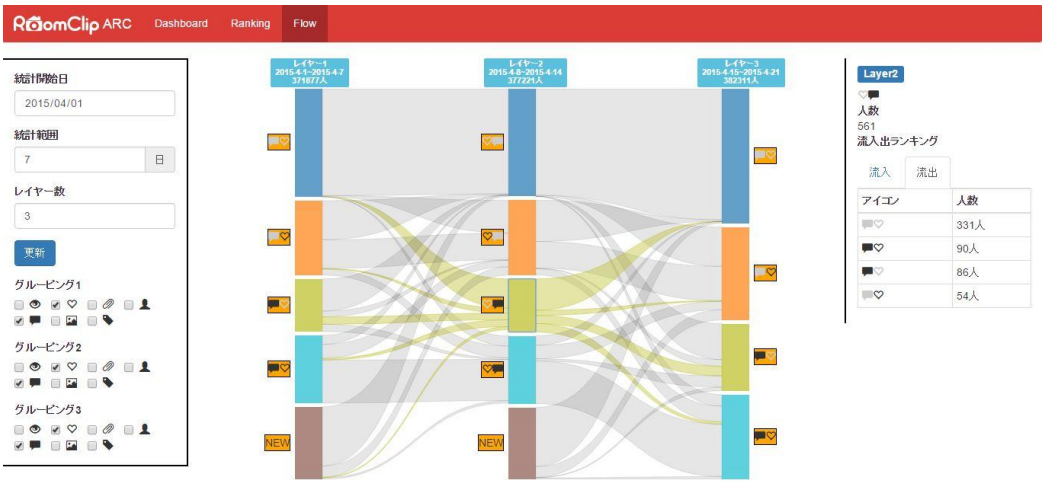


図 7.1: Flow 画面におけるグルーピングの例

このようにグルーピング操作を行うことで、ある特定のアクションの相関をみることができる。その相関の結果から施策に反映できるインセンティブを発見することが、Flow 画面の狙いである。この例では like と comment に何らかの相関があると踏んでグルーピングを行ったが、このように like と comment の相関に当たりをつける根拠が実際には中々見つからない。

そこで因子分析を行う。因子分析とは、多変量データに潜む共通因子を探り出すための手法である。因子分析を用いることで、ユーザアクティビティをいくつかの因子に要約し、アクションの相関を具体的な数値で表す。この結果を用いることで、Flow 画面をより効果的に使うことができるようになる。

7.2 因子分析モジュールの実装

API として実装するためにプログラミング言語として python を用い、因子分析には R 言語を用いた。ライブラリとして pyper を使うことで python 上で R を実行している。R 上ではパッケージとして psych を用いている。以下にバージョンを示す。

名称	バージョン
python	3.4.3
R	3.2.2
pyper	1.1.2
psych	1.5.8

表 7.1: 因子分析モジュールの開発環境のバージョン

7.3 API の仕様

アクティブユーザの因子分析モジュールを API に組み込んで ARC に実装した。この API はパラメータとして日付（YYYY-MM-DD）を指定すると、その日からの 1 週間の各ユーザのアクティビティの累積数を対象に因子分析を実行し、その結果を返す。日付に加えて分析期間を週の数としてパラメータに追加すると、指定されて日付から指定された週の期間を対象として因子分析を実行する。例えば、1 月 1 日と 3 週間がパラメータとして指定された場合、1 月 1 日から 1 月 7 日、1 月 8 日から 1 月 14 日、1 月 1 日から 1 月 7 日までの 3 週間分のユーザのアクティビティの累積数を対象として因子分析を実行する。

API 上では次のように動作している。まず指定された日付から 1 週間分のデータを user_activity テーブルから取得する。続いてそのデータを集計して、1 週間に各ユーザが行った 7 つのアクションそれぞれの累計を算出する。（週の期間が指定されている場合は、ここでその週の分だけ集計を行う。）そのデータを R に渡し、R 上で因子分析を行う factanal() 関数を実行する。因子数は 3 で、回転軸は promax 回転としている。この 2 つの要素は [16] や [17] を元に決めた。factanal() 関数の実行結果を python に渡し、その値を出力している。

7.4 実行例と考察

因子分析の実行例を次の表にまとめる。対象としたデータの期間は、Tunnel 社から頂いたデータの全期間である。

view	like	clip	follow	comment	post	favtag
0.609	0.487	0.863	0.005	0.368	0.609	0.937

表 7.2: 因子分析の実行例：独自因子

	Factor1	Factor2	Factor3
view	0.228	-0.037	0.496
like	0.661	0.018	0.090
clip	-0.152	-0.020	0.427
follow	-0.013	1.001	0.001
comment	0.824	-0.046	-0.034
post	0.540	0.071	0.102
favtag	-0.111	0.063	0.260

表 7.3: 因子分析の実行例：因子負荷量

	Factor1	Factor2	Factor3
寄与率	0.214	0.145	0.074
累積寄与率	0.214	0.359	0.432

表 7.4: 因子分析の実行例：因子間の寄与率

独自因子とは、因子に対する変数の独自性を表し、1 に近づくほど独立の度合いが高い。因子負荷量とは、各因子が変数に及ぼす影響の度合い。0.3 以上あれば影響があるとみなせる。

第 1 因子は like、comment、post を多く行い、view もほどほど行うユーザ——つまり、他者と積極的にコミュニケーションを取るユーザを表している。第 2 因子は follow だけを行うユーザを表している。第 3 因子は view、clip、favtag をほどほど行うユーザ——つまり、他者に興味はなく写真に興味があるユーザを表している。これらの因子の累積寄与率は 0.432 と半分にも満たず、また view と clip、post、favtag の独自因子の高さから、ユーザのアクティビティはこれら 3 つの因子では 5 割程しか説明できていないことになる。つまり、全期間を通したユーザのアクティビティには目立った特徴がないことが確認できた。

	Factor1	Factor2	Factor3
Factor1	1.000	0.314	0.378
Factor2	0.314	1.000	0.492
Factor3	0.378	0.492	1.000

表 7.5: 因子分析の実行例：因子間の相関係数

第8章　既存の分析ツール

分析システム ARC を開発するにあたって、いくつかの汎用データ分析ツールとサービス特化型分析ツールを調査した。ここでは筆者が調査した 4 つの分析ツールについて述べる。

8.1　Google Analytics

Google Analytics[18] は、Google が提供しているアクセス解析ツールである。「サイト訪問者の数」「1 ページ当たりの訪問数」「1 ページ当たりの離脱数」「1 ページ当たりの滞在時間」「どこからアクセスしてきたのか」「何のデバイスからアクセスしてきたのか」など、サイトへのアクセスに関するさまざまなデータについて詳しく分析することができる。Google アカウントを取得すれば、基本的に無料で利用できる。

Google Analytics は Tunnel 社も導入している。そこで開発した分析システム ARC では、ユーザが RoomClip のページを閲覧するアクション――view のデータをベースとし、view が行われた上でさらにどのようなアクションを行っているか、というようなユーザの行ったアクションについて分析を行った。

8.2　Rite Tag

Rite Tag[19] はハッシュタグ分析ツールである。Twitter、Facebook、Google+に対応している。このツールはタグのポピュラー具合を色で分けて表示し、一緒に使われるタグのリコメンドを提示する。その他の機能としては、リツイートや返信、お気に入りといった他ユーザからの反応をもらい投稿内容のリコメンドや、投稿の予約などの機能がある。

RoomClip でも投稿する写真にタグをつけることができ、またタグをお気に入りに登録するアクション――favorite-tag がある。開発した分析システム ARC では favorite-tag のアクション数も分析対象とした。開発当初はタグの分析も行う予定であったため、このようなタグに特化した分析ツールも調査していた。

8.3　Social Insight

Social Insight[20] はソーシャルメディア分析ツールである。Twitter、Facebook、Instagram、Google+などに対応している。特徴としては主に 3 点あげられる。まず 1 点目は口コミ傾聴分析である。Twitter のツイート内容を過去 3 年分以上に渡って分析したり、自社ブランドに興

味を持っているユーザ層を把握したりすることができる。2 点目は SNS アカウント分析である。競合するアカウントの分析や、エンゲージ率の時間帯分析ができる。3 点目は投稿予約管理である。複数のメディアの投稿を一括で管理できる。

RoomClip のユーザは投稿された写真についてコメントを残すことができ、それが RoomClip 上の多様なコミュニティに繋がっている。このコメントの形態素解析を行うことで、RoomClip 上のコミュニティの動きを読むことができると考えたが、今回はそこまで実装することはできなかった。

8.4 Minitab17

Minitab17[21] は、有料の統計解析ソフトウェアである。シンプルなインターフェースと直感的な操作性を兼ね備えている。主な機能としては、秘儀ストグラムや散布図などのグラフ分析、回帰分析や分散分析などの基本統計、管理図や工程能力分析などの統計的品質管理、要因計画やタグチ計画などの実験的計画法などがある。

分析をする度にデータをフォーマットに合わせて加工してインポートするというのは、非常に煩雑である。開発した分析システム ARC では、非アクティブユーザの予測やユーザアクティビティの因子分析がデータのインポートなしで、API として簡便に利用できる。

第9章　評価

本プロジェクトでは第二次開発を終えた後に試用期間を設けて、実際に Tunnel 社に Ranking 画面と Flow 画面を使ってもらい、フィードバックを頂いた。本章では頂いたフィードバックとそれに対する考察を述べる。

9.1　フィードバック

まず Ranking 画面に関しては、「アクティビティを促進させる」施策を打った場合、どの程度変動するのかを期間登録者ベースでみることで効果測定として有効である、との評価を頂いた。つまりこの画面では「標本を絞り込んだ理由」または「期間を絞り込んだ理由」の分析ができる、とのことであり、具体的には以下の 2 要素になる。

1.ある期間にキャンペーンを実施した場合、その効果の有無の確認

2.ある特殊な経路で入ってきた登録者について、彼らの変遷の経緯の確認

一方でそのような使い方をする場合、「どういうユーザの特性があるのか」といったような、現状分析にはあまり向いておらず、何の施策を打っていない状態での仮設立脚には至らなかった、とのコメントも頂いた。

Flo w画面に関しては、「ある行動をしたユーザのその後の行動」ではなく、「ある行動をしたユーザのその前の行動」を調べるために用いた、とのコメントを頂いた。つまり「ある行動をさせたい」と思ったときのトリガーを分析していたようである。また、非アクティブユーザがアクティブユーザになるトリガーも分析していたようである。分析の結果としては、施策に結びつくほどの分析はできなかったが、逆に「顕著な傾向がない」ということが確認できた、とのコメントを頂いた。

9.2　考察

Ranking 画面に関しては狙い通り、施策の効果を測定するような画面を作れたように思う。この画面を更に改良するとすれば、ノードの詳細情報の表示が挙げられる。というのも、ARC ではユーザを 7 つのアクション行ったことがあるかないかで 65 コのノードに分類しているが、この行ったことがあるかないかの 2 値で分類しているところに限界がある。3.2 で述べたよう

に、ユーザを大きな意味合いを持つクラスタに分類し、あるアクションを行っていないユーザを検出するために、我々はこのような 2 値分類をした。しかし一方で、「アクションを行っている」という状態には、1 回行った状態と 2 回行った状態の両方が含まれていることを指す。現状は「0 か 1 以上か」の 2 値なので、次は「0 か 1 か 2 か...n か」のようなより細分化した分析を行いたい。例えば、「view あり、like あり、clip なし、follow なし、comment なし、post なし、favtag なし」のようなノードでは、今までの変動人数などの情報の他に、view の回数と ike の回数の平均や分散を示す。このようにすることでより具体的に施策の効果を測定できるようになるだろう。

Flow 画面では、「顕著な傾向がなく、施策の仮設立脚までには至らなかった」とのコメントを頂いた。「顕著な傾向がなかった」ということは、顕著な傾向の可視化という意味では Flow 画面の狙い通りではあったが、実際に使ってみると「顕著な傾向がなかったことが確認できた」というのは非常に残念である。顕著な傾向があるかどうかはユーザのデータに依存するので、やりようがない部分ではある。しかし今回はユーザのアクションの回数にのみ注目しているので、回数以外のデータに着目することで、顕著な傾向を見つけることができるかもしれない。具体的には、like や comment をする写真、favtag をするタグ、コメントログの分析などが考えられる。ユーザの傾向の抽出には、このようなより一歩踏み込んだ分析が必要なのだろう。

第10章　振り返り

本章では、プロジェクト全体を振り返っての筆者の所感を述べる。

まずプロジェクトの発足当初は、プロジェクトの目的が非常に曖昧なまま議論を重ねることが多く、話は一進一退の繰り返しだった。中間発表1までに要件をある程度の形にまとめあげ、発表後にTunnel社との打ち合わせを行ったが、Tunnel社の要望と我々の要件は少し異なっていた。我々はユーザのアクティブ率がそれなりにある前提で、そこからさらにRoomClipのサービスを向上させるための要件を考えていたが、彼らはユーザのアクティブ率に満足しておらず、まずはアクティブ率を増やしたいと言っていた。思うに、やはり早い段階でTunnel社の現状を彼らから直接伺うべきだったと考えている。とは言え、その要件をまとめるために行った全ての作業が無駄だったわけではなく、その過程でRoomClipというサービスについて深く考えることができたからこそ、ARCというシステムを提案することができた、とも考えている。

打ち合わせを終えて方向性がある程度固まってからは、役割を分担して開発を始めた。開発自体は課題担当教員である岡先生、橋本先生のご助力のおかげで特に大きな問題もなく順調に進行した。その後の2回目の打ち合わせでは実際にARCのプロトタイプを触ってもらい、いいフィードバックを得ることが出来た。そのフィードバックをもとに更なる開発に取り組み、3回目の打ち合わせに臨んだ。3回目の打ち合わせの後に試用期間を設けて、ARCへのフィードバックをドキュメントとして頂いた。しかし今思えば2回目の打ち合わせの時点で試用期間を設けて、その時にもドキュメント形式のフィードバックを彼らから頂くべきだったかもしれない。というのも、12月の試用期間後のフィードバックは2回目のそれと比べて、よりARCの施策への貢献の度合いについて詳細に述べたものだったからである。このフィードバックの内容の差は、当然ARCの完成度の違いが大きな要因となっているだろうが、試用期間を定めて時間をかけてフィードバックを頂くというのはとても重要なことだと感じた。

通しての所感としては、顧客の要望の抽出にせよ、システムの試用にせよ、もう少し先を見据えた速いタイミングでの行動ができれば、よりスムーズにプロジェクトを推進できたように思う。

第11章　結論

本プロジェクトは、Tunnel 株式会社を顧客に、「写真共有サービスのユーザ動向可視化・分析システムの開発」をテーマとして実施した。

Tunnel 社が運営している RoomClip はインテリアに特化した SNS である。RoomClip に登録することでユーザは自分の好きな部屋やインテリアの写真を投稿することができ、さらに投稿した写真上で他のユーザとコミュニケーションを取ることができる。このため RoomClip ではユーザのアクセス数を増加させて投稿写真数を増加させることがサービスの維持、ひいては発展に繋がる。しかし現状、RoomClip のユーザのアクセス率（1 週間に 1 回以上アクセスしたことのあるユーザの割合）は、5%〜10%にとどまっている。そこで Tunnel 社から我々に対して、サービス活性化のために「ユーザの行動の変化を詳細に把握し、施策がユーザの行動に与えている影響を知りたい」との要望があった。要望を実現するために、我々はユーザの動向の分析及び可視化を行うシステム—— ARC (Analysis for the RoomClip) ——の開発を行った。

RoomClip には view、like、clip、follow、comment、post、favorite-tag という 7 つのアクションがある。ユーザの動向を分析するにあたって、それぞれのユーザが 7 つのアクションそれぞれ行ったことがあるかないかの 2 値で分類することにした。分類の数は 65 コとなり、これをノードと呼ぶ。

ARC には 3 つの画面と 2 つの API がある。3 つの画面とは、ノードに所属するユーザの変動人数を示す Ranking 画面、ノード間のユーザの流出入を示す Flow 画面、7 つのアクションの利用率を示す Dashboard 画面のことであり、2 つの API とは非アクティブユーザを予測する API、ユーザアクティビティの因子分析を行う API のことである。

筆者は ARC の開発において、データベースの設計・実装、非アクティブユーザを予測するモデルの実装、ユーザアクティビティの因子分析の実装を担った。非アクティブユーザを予測するモデルの実装では、分類器によって大きな性能の差は見られなかったが、変数の重要度が大きく変わることが確認できた。ユーザアクティビティの因子分析の実装では、他者と積極的に関わるユーザや follow だけをするユーザ、写真に興味があるユーザがいることが確認できた。

Tunnel 社に Ranking 画面と Flow 画面を試用して頂いた。評価としては、Ranking 画面は「施策の効果測定として有効」であり、Flow 画面は「施策に結びつくほどの分析はできなかったが、顕著な傾向がないことが確認できた」とのコメントを頂いた。施策に結びつけるような分析を行うには、アクションの回数の分析のみならず、like や comment をする写真、favtag をするタグ、コメントログの分析をする必要があると思われる。

謝辞

本プロジェクトを遂行して特定課題研究報告書を執筆するにあたり、非常に多くの方のお世話になりました。この場を借りて感謝の意を表したいと思います。

顧客である Tunnel 株式会社の皆さまにはテーマの提供並びにご指導、ご協力を頂けました。誠にありがとうございました。

指導教員である田中二郎先生にはプロジェクトの遂行並びに中間報告書の添削、特定課題研究報告書の執筆において多くのご指導を頂きました。誠にありがとうございました。

課題担当教員である岡瑞起先生、橋本康弘先生には、数多くのミーティングに参加して頂き、分析に関する専門的な知識やシステムに関する技術から中間発表のプレゼンや特定課題研究報告書の執筆に至るまで多くのご指導を頂きました。誠にありがとうございました。

最後に、本プロジェクトのチームメンバである、古谷氏、崔氏、万氏とは多くの議論を重ねることで、非常に有意義なプロジェクト活動をすることができました。誠にありがとうございました。

参考文献

[1]RoomClip: 「RoomClip」 http://roomclip.jp/ (2016/01/09)

[2]Tunnel: 「企業様お問い合わせ」 http://roomclip.jp/doc/official (2016/01/09)

[3]SEO SCENE: 「コホート分析」 http://seo-scene.com/cohort-analytics/ (2016/01/09)

[4]Mike Bostock: 「Sankey Diagrams」 http://bost.ocks.org/mike/sankey/ (2016/01/09)

[5]Martin Fowler: 「microservice」 http://martinfowler.com/articles/microservices.html (2016/01/09)

[6]Amazon Web Service: 「Redshift」 https://aws.amazon.com/jp/redshift/ (2016/01/09)

[7]Richard J. Oentaryo, Ee-Peng Lim, David Lo, Feida Zhu and Philips K. Prasetyo: Collective Churn Prediction in Social Network. Proc. IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining(ASONAM2012), IEEE Computer Society, pp.210-214(2012)

[8]Xi Long, Wenjing Yin, Le An, Haiying Ni, Lixian Huang, Qi Luo and Yan Chen: Churn Analysis of Online Social Network Users Using Data Mining Techniques. Proc. International Multi Conference of Engineers and Computer Scientists(IMECS2012), IMECS, pp.551-556(2012)

[9]Blaise Ngonmang, Emmanuel Viennet and Maurice Tchuente: Churn prediction in a real online social network using local community analysis. IEEE/ACM Conference on Advances in Social Networks Analysis and Mining(ASONAM), pp.282-288(2012)

[10]Yaya Xie, Xiu Li, E.W.T. Ngai and Weiyun Ying: Customer churn prediction using improved balanced random forests. Expert Systems with Applications 36(2009), pp.5445-5449

[11]Jaya Kawale, Aditya Pal and Jaideep Srivastava: Churn Prediction in MMORPGs: A Social Influence Based Approach. Computational Science and Engineering, 2009. CSE '09. International Conference on (Volume:4), pp.423-428

[12]土井千章, 川崎仁嗣, 中川智尋, 片桐雅二, 稲村浩, 太田賢: 決定木を用いたソーシャルネットワークサービス向けのアクティブユーザ予測モデルの提案. マルチメディア, 分散, 協調とモバイル (DICOMO2014) シンポジウム, pp.366-372

[13]Tianqi Chen and Tong He: Higgs Boson Discovery with Boosted Trees. JMLR: Workshop and Conference Proceedings 42, pp.69-80, 2015

[14]Continuum Analytics, Inc.: 「Anaconda」 https://www.continuum.io/why-anaconda (2016/01/09)

[15]石井康夫, 油井毅, 竹安数博: SNS に対する利用者意識の分析. 国際研究論叢 26(2) : 1 - 21, 2013

[16]堀啓造: 因子分析における因子数決定法. 香川大学経済論叢 第 77 巻 第 4 号 2005 年 3 月 35-70

[17]SlideShare: 「R で因子分析」 http://www.slideshare.net/simizu706/r-42283141?related=1 (2016/01/09)

[18]Google: 「Google Analytics」 http://www.google.com/intl/ja_ALL/analytics/index.html (2016/01/09)

[19]Saul Fleischman: 「Rite Tag」 https://ritetag.com/ (2016/01/09)

[20]User Local: 「Social Insight」 http://social.userlocal.jp/ (2016/01/09)

[21]構造計画研究所:「minitab17」http://www2.kke.co.jp/minitab/products/minitab/ (2016/01/09)

付録

付 録A　Tunnel株式会社からのフィードバック文書全文

試用期間後に頂いた Tunnel 株式会社からのフィードバック文書全文を示す。

ARCフィードバックシート

はじめに

サービスURLにあるRoomClip分析ツール「ARC」を使って、フィードバックを追記していくシートです。何か新しい発見があり次第随時追記していきます。

サービスURL

http://arc-front.tk/

Tunnelフィードバック

Rankingに対するフィードバック

どのような操作をし、どう思ったか

＊単純にどういう行動している人がいっぱいいるのかを確認した
　　└今まで「いいね」何人、「クリップ」何人という単独でみていたが、それがベン図のように見えたため、「主にどういう行動をユーザーがしているのか」が見渡せた
　　└ただし相関がみれるわけではないので、それ以上の分析・インサイトはなかった

＊ある日の登録者を絞り込み、登録日から1日毎に期間をずらして変動率をおっかけた
　　└初日からアクティビティは減る一方であることがわかった
　　└あるタイミングでアクティビティが増えるという状況がもしあるならば、例えば「本当は撮影するユーザーでも、10個くらいいいねしたりコメントしたり様子を見たあとに投稿する」ようなことが言えるかもしれない、と考えた。

＊ある期間に登録したユーザーのアクティビティがどう減少していくのかをデイリー単位でみようとしたが、できなかった
　　└きっと減少の速度がアクティビティによって変わるはずだ、という想定があった

＊どの行為をやっているユーザーが最後までのこる（ゼロにならない）のかをみた
　　└スティッキーに使っているユーザーがどのアクティビティをやっているのかを確認したかったが、結構変動が激しく、捉えられなかった

＊ランキングの推移を確認しようとした
　　└ある時に登録した人のランキング変位がどのくらいのタイミングでおこるのかを見ようとしたが、煩雑でできなかった

総評

「なにもない状態で分析しろ」といわれたら、やはりフローについて見たがる傾向にあると自己認識した。ただし、もし「いいね促進キャンペーン」などを特定の期間ではっていた場

図 A.1: フィードバック文書 1

合、どの程度変動するのかを期間登録者ベースでみれると、効果測定としてはとてもよいと感じた。
結果、ランキング機能は「標本の絞り込み」と「期間」が非常に重要で、
つまり「標本を絞り込んだ理由」か「期間を絞り込んだ理由」が分析の要になると思った。
具体的には下記2つ。

1. ある期間にキャンペーンを張った場合、それが効果があったのか？の確認
2. ある特殊な経路で入ってきた登録者の場合、それがどのように変遷したのか？の確認

のようなことに効果的だと言えそうだ。
その場合これは「どういうユーザー特性があるのか」といったような、現状分析にはあまり向いておらず、施策の評価として意味をもつ、ということなので、なにも施策をうっていない状態での仮説立脚には至らなかった。

Flowに対するフィードバック

どのような操作をし、どう思ったか

＊ある時写真投稿をした人の、それより前のアクティブ履歴を眺めた
　　└母数が多いため「いいね」によってしまっているが、「なにもしない」から突如写真を撮る人もいるということもわかり、非常に興味深かった
　　└ただし「黄金パターン」のようなものはわからず。（そもそもないのかも）

＊クリップした人とそうでない人で、後の行動に変化がおきるかをみた
　　└流出のランキングをみたくなったが、一応手作業でみれた。が、クリップした人のほうが有意に撮影する、みたいな情報までは辿りつけなかった

＊ある時何もしなかった人（見る、とそれ以外で絞り込んだ）がその後どうしたのか、またはその前何をしていたのかを眺めた
　　└特に傾向がつかめなかった…

総評

ほとんど3層分析をして、常に中央または右端の層を固定して調査していた。
つまり、「ある行動をしたユーザーのその後の行動」を気にしたのではなく、
「ある行動をしたユーザーのその前の行動」を気にしていた。
「ある行動をさせたい」と思った時、どのツボを刺激すればしてもらえるのか、を求めて色々と出力をしていた。
逆に「ある行動をしなくなった」人のみ、「その後どんな行動をしたのか」をみた。
休眠ユーザーがどんなアクティビティフックで起き上がるのか、を調べたかった。
結果、本質的には「ユーザーがアクティブになるために必要なアクションを分析しようとした」という分析姿勢だったと思われた。
施策に結びつくほどの分析はできなかったが、逆に言うと「顕著な傾向がない」ということがわかった。

開発要望

図 A.2: フィードバック文書 2

- ランキングの日付境界ロジックが、登録者絞り込みと期間1,2でことなるを統一して欲しい。（2015-12-01 ~ 2015-12-02 で12月1日分なのだが、期間だと 2015-12-01 ~ 2015-12-01で12月1日分） → 対応しました。2015-12-01 ~ 2015-12-01で12月1日分になるように統一しました。（古谷）
- ランキングで更新を押した時に、ソート条件を同時に更新して欲しい。→ ソート条件は更新後も反映されているはずです。（古谷）
- フローで流入・流出をソートしたい。→ 対応しました。（崔）
- フローで流入・流出の出元・出先ベースでの割合がでていて欲しい。
- フローで「何もしなかった人」グループを選択したい → グルーピング時にVIEWのアイコンをチェックすると、VIEWしていないグループが選択できるようになります。そのグループが「何もしなかった人」グループです。灰色にして強調することを検討中です。（崔）

図 A.3: フィードバック文書 3

付 録B　ユーザシナリオ一覧

ARC を作る際に考えたユーザシナリオを示す。

古谷
1.
　ある日、平山さんがTunnel社に出社し、Google Analytics ToolでRoomClipのページビューを確認すると、先週よりもPV数が増えていることに気がついた。
　なぜか考えた時に、「RoomClipのコメントが炎上しているのか？」「雑誌がブログで取り上げられたのか？」「Twitterで大量にRTされているのか？」などといろいろと原因を考えたが、どれが決定的な原因かはわからない。
　そこで、筑波大学の学生が開発したARCを立ち上げ、ランキング画面を開いてみた。先週１週間と今週１週間の違いを見るため、スパンを7に設定し、先週の日付と今週の日付を入力し、ランキングを表示した。次に、何のアクションを起こした人が増えたのかをとりあえず確認するため、変動人数が多い順でソートし、それぞれのアクションごとでどれくらい人数が増えたのかを１つずつ確認していった。
　すると、COMMENTのアクションを行った人が特に増えていることに気がついたので、RoomClip内のコメント機能の中で何かしら炎上が起きている可能性があるということが推測された。
（そこで実際にどの写真が炎上しているのかを確認して、特定のタグがついた写真が炎上していることがわかったので、それに対して対処を行うことが出来た。）

2.
　ある月末、平山さんは、先月にRoomClipに追加した「ユーザが写真を投稿することを促す機能」の効果を測定するため、先月と今月で写真の投稿者数がどれくらい増えたかを確認することになった。
　そこで、ARCを立ち上げ、先月の1日と今月の1日を入力し、スパンを1ヶ月に設定した。さらに、グルーピング機能でPOSTにチェックを入れ、POSTしている人としていない人の2種類にユーザを分割し確認してみた。
　すると、POSTしている人もしていない人も増えてはいたが、POSTしている人のほうが変動率が高いことがわかり、写真を投稿しているユーザが増加傾向にあることが確認できた。

万
1、RoomClipのサービスが立ち上げてからN周年記念のため、Tunnel社が一ヶ月の期間のイベントを開催する。イベント期間中、利用者が投稿した写真数が１０枚以上、一枚あたりの写真が得られた「Like」は３０人以上のユーザーに対し、アンケートを配り、アンケートに記入した方は、Tunnel社からハガキや家具メーカーのクーポンをもらえる。このようなイベントに伴う、平山さんはRoomClipの写真の投稿数と「Like」を押した回数がはどれぐらい増えたかを確認するため、ARCを立ち上げ、イベントが開催された一ヶ月の期間を指定し、グルーピング機能でPOSTとLIKEにクリックを入れ、この一ヶ月間に、投稿数と「Like」数は大量に増えた、その中「Like」数より投稿数の変動率が大きいことがわかり、イベントの開催により、利用者のアクティビティーをあげられた。

２、同じくインテリアをシェアするウェブサービスである「HouseClip」は最近ますます人気になったと平山さんが気がづいた、「うちの利用者が奪われるか」「HouseClipのせいで、新規登録者が減るか」と平山さんは心配になった、そこで「ARC」を立ち上げ、最近RoomClipの運営状況をチェックする、FLOWで９月１日の前後３０日のスパンを指定し、９月１からの３０日間の新規登録者の人数が増えたが、変動範囲は先月より小さくなったことがわかった。「HouseClip」は「RoomClip」に影響をあたえられたと判定できる。

図 B.1: ユーザシナリオ 1

３、９月からRoomClipのコンメント機能にスタンプを貼りつる機能を追加した。利用者はコメントするときにRoomClipが提供するスタンプを入力できるようになった。平山さんは、この新しい機能の追加に伴う、利用者たちのコメント数が増えたか確認するため、ARCを立ち上げ、９月1日から９月３０日のスパンを指定し、さらに、グルーピング機能でCOMMENTにクリックを入れ、FLOW図からコメントした人が増えたことがわかり、スタンプ機能の追加により、利用者のコメント意欲を促せることを確認できた。

崔
背景
「クリップ機能は何なのかよくわからない」という書き込みを2chのRoomClip版で発見した。そこで運営側は、クリップ機能の利用率向上を狙って、「クリップを使ってみよう」とのバナーを掲載した。
「クリップ機能は何なのかよくわからない」という書き込みを2chのRoomClip串で発見した。そこで運営側は、クリップ機能の利用率向上を狙って、「クリップを使ってみよう」とのバナーを掲載した。
問題点
一ヶ月経った時点で、クリップ機能の使用状況を分かりやすいグラフでみたい
一ヶ月経った時点で、クリップ機能の使用状況を分かりやすいグラフでみたい
問題解決
1.ARCのflowページの開いて、バナー掲載前の一ヶ月とバナー掲載後の一ヶ月のフローをみる
2.「clip」でグルーピング
3.ポインターをブロックの上に移動することで、宣伝前後の、clip機能の絶対人数が分かる
4.前の一ヶ月で「clip」機能を使ったユーザ（8000人）は、後の一ヶ月もclip機能を使ったユーザは6000人（75%）であることが分かる（ブロックをクリックして、流出線の上にポインターを移動）
5.前の一ヶ月で「clip」機能を使わなかったユーザ(30000人)の中で、後一ヶ月でclipを使ったユーザは10000人であることが分かる（上と同様）

●河越
Tunnel社はある施策を行い、ある程度まとまった数の新規ユーザの獲得に成功した。今まで新規ユーザの獲得には難航していたので、今回獲得した新規ユーザには何としてもRoomClipを使い続けてもらいたい。そこで今回獲得した新規ユーザの内の何人が来週もRoomClipを使い続けてくれるかどうかを調べ、来週も使い続けてくれるユーザが少ないのであれば明日明後日にでも新たな施策を実施したい。この要求を解決するために機械学習システムを利用する。AWSで自動的に取得している今週のユーザのアクティビティのログを入力として与えることで、そのユーザが来週もRoomClipを使い続けるかどうかがわかる。今回獲得した新規ユーザのアクティビティのログを入力すると、その内の７割は来週は使わなくなるという結果が出力された。そこでTunnel社は新規ユーザにRoomClipを長く使い続けてもらうために新たな施策を打とうとミーティングを行うのだった。

図 B.2: ユーザシナリオ 2

付 録C　サーバ設定書

フロントサーバとバックサーバの設定書を示す。

C.1　フロントサーバ

C.2　バックサーバ

フロントサーバ設定書
設定者：古谷

参考
http://blog.genies.jp/2011/08/amazon-ec2-amazon-linux.html
http://itpro.nikkeibp.co.jp/article/COLUMN/20080219/294153/
http://www.maruko2.com/mw/%E4%B8%80%E8%88%AC%E3%83%A6%E3%83%BC%E3%82%B6%E3%83%BC%E3%82%92_sudo_%E3%81%A7%E3%81%8D%E3%82%8B%E3%82%88%E3%81%86%E3%81%AB%E3%81%99%E3%82%8B
http://dev.classmethod.jp/etc/amazon_linux_nginx_basic_auth/

サーバ構築
Amazon EC2 で無料枠でインスタンスを作った
　　********@gmail.com で新しくアカウントを作った

OS は Amazon Linux を選択、ほかの設定はいじってない

キーペアは　picture01front_furuya.pem → furuya アカウントの ppk と同じ名前

　　IP は　********

サーバ設定

キーペアを利用し ec2-user でログイン後、furuya のアカウントを作り、秘密鍵でログインできるようにした
--
$sudo su -
#useradd -d /home/furuya -m furuya
#cd /home/furuya
#mkdir .ssh
#ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /home/furuya/.ssh/id_rsa
Enter passphrase (empty for no passphrase):　（パスフレーズはなしにした）
Enter same passphrase again:

図 C.1: フロントサーバの設定書

バックサーバ設定書
設定：古谷

参考
http://kazsoga.com/amazon-linux-python3-install/
http://symfoware.blog68.fc2.com/blog-entry-1412.html
http://qiita.com/moiwasaki/items/57f0f3382ca580c3b6d5
http://yakiniku.hatenablog.com/entry/2015/01/31/192757

Anaconda
Python3 のアンインストール（入ってた場合）
/etc/profile の最終行にあった PATH 設定文（$ export PATH=$PATH:/opt/ptyhon3/bin）を削除
Anaconda のインストール用.sh ファイルをダウンロードし実行
インストール先は /opt/anaconda3 を指定
$ export PATH="/opt/anaconda3/bin:$PATH" を /etc/profile に追加
pip でライブラリを再インストール
pyramid
arc-back を使うときは以下も同時に
$ python setup.py develop
$ initialize_arc-back_db development.ini
pit
$ sudo patch -u /opt/python3/lib/python3.4/site-packages/pit.py pit_py3.diff を忘れず
psycopg2

R
$ sudo yum install -y R git libxml2-devel openssl-devel libcurl-devel libjpeg-devel libpng-devel mysql-devel

PypeR
$ sudo pip install pyper

PostgreSQL をインストール
path を設定

図 C.2: バックサーバの設定書

付 録D　API仕様書

ARC の API の仕様書を示す。

API仕様（v3.3）

作成者：古谷翔太
更新日：2015-12-21（Dashboardの設計を変更）

GET
~/api/v3/ranking/{start1}/{start2}/{span}

今日から2週間前〜1週間前と1週間前から今日までを集計したランキングを返す

パラメタ名	説明
start1	起点となる日付1 (YYYY-MM-DD)
start2	起点となる日付2 (YYYY-MM-DD)
span	集計期間（起点となる日付を含める）

GET
~/api/v3/ranking/{start1}/{start2}/{span}/registration/{startreg}/{endreg}

指定した期間に新規登録したユーザに関して集計したランキングを返す

パラメタ名	説明
startreg	新規登録期間開始日
endreg	新規登録期間終了日
start1	起点となる日付1
start2	起点となる日付2
span	集計期間（起点となる日付を含める）

GET ~/api/v3/flow/{start1}/{start2}/{span}

集計したノード間流出入を返す

パラメタ名	説明
start1	起点となる日付1
start2	起点となる日付2
span	集計期間（起点となる日付を含める）

図 D.1: API の仕様書 1

GET
~/api/v3/flow/{start1}/{start2}/{span}/registration/{startreg}/{endreg}

指定した期間に新規登録したユーザに関して集計したノード間流出入を返す

パラメタ名	説明
startreg	新規登録期間開始日
endreg	新規登録期間終了日
start1	起点となる日付1
start2	起点となる日付2
span	集計期間（起点となる日付を含める）

GET ~/api/v3/flow/{start1}/{start2}/{span}/all

集計したノード間流出入を返す＜NodeID65(noview), 66(new) 含む＞

パラメタ名	説明
start1	起点となる日付1
start2	起点となる日付2
span	集計期間（起点となる日付を含める）

GET ~/api/v3/transition/actions

全ユーザを対象とし、全期間の日付ごとに全ユーザ数と各アクションを行った人数を返す

GET ~/api/v3/transition/actions/{startreg}/{endreg}

startregからendregの間に登録したユーザを対象とし、全期間の日付ごとに全ユーザ数と各アクションを行った人数を返す

パラメタ名	説明
startreg	ユーザの登録範囲開始日
endreg	ユーザの登録範囲終了日

GET
~/api/v3/transition/actions/{startreg}/{endreg}/{start}/{end}

startregからendregの間に登録したユーザを対象とし、startからendまでの期間の日付ごとにユーザ数と各アクションを行った人数を返す

図 D.2: API の仕様書 2

パラメタ名	説明
startreg	ユーザの登録範囲開始日
endreg	ユーザの登録範囲終了日
start	集計する範囲の開始日
end	集計する範囲の終了日

GET ~/api/v3/transition/registrations
全期間の日付ごとの登録者の推移を返す

GET ~/api/v3/transition/registrations/{start}/{end}
startからendまでの日付ごとの登録者の推移を返す

パラメタ名	説明
start	範囲の開始日
end	範囲の終了日

==

◆rankingのJSONフォーマット

```
{
	start1：2015-07-20, // 起点となる日付1
	start2：2015-07-27, // 起点となる日付2
	span：7, // start1,2からの集計期間(start1,2も含める)
	ranking[65]： [ {
		nodeid：4, // NodeID（下記参照）
		actions：[“VIEW”, “LIKE”, “POST”], // Actions（下記参照）
		range1：2000, // start1に関する集計の結果
		range2：2200 // start2に関する集計の結果
	} ]
}
```

◆flowのJSONフォーマット

```
{
	start1：2015-07-20, // 起点となる日付
	start2：2015-07-27, // 起点となる日付2
	span1：7, // start1,2からの集計期間(start1,2も含める)
	nodes[65]： [ {
```

図 D.3: API の仕様書 3

```
        nodeid：4, // NodeID（下記参照）
        actions：[“VIEW”, “LIKE”, “POST”], // Actions（下記参照）
        range1：2000, // start1に関する集計の結果
        range2：2200 // start2に関する集計の結果
        flowout[65]：{ // 各ノードに対して出ていく数に関する情報
            nodeid：2, // NodeID
            actions：[“VIEW”, “COMMENT”], //Actions
            num：100 // 出ていく数
        }]
}
```

◆flow/allのJSONフォーマット

```
{
    start1：2015-07-20, // 起点となる日付
    start2：2015-07-27, // 起点となる日付2
    span1：7, // start1,2からの集計期間(start1,2も含める)
    nodes[66]： [ {
        nodeid：4, // NodeID（下記参照）
        actions：[“VIEW”, “LIKE”, “POST”], // Actions（下記参照）
        range1：2000, // start1に関する集計の結果
        range2：2200 // start2に関する集計の結果
        flowout[66]：{ // 各ノードに対して出ていく数に関する情報
            nodeid：2, // NodeID
            actions：[“VIEW”, “COMMENT”], //Actions
            num：100 // 出ていく数
        }]
}
```

◆transition/actionsのJSONフォーマット

```
{
    start: 2014-01-01, // 開始日
    end: 2015-01-01, // 終了日
    startreg: 2014-12-01 // 登録開始日
    endreg: 2014-12-01 // 登録終了日
    data：[{
        date: “2015-04-01”, // 日付
        views: “50” // 以下、この日に各アクションを行ったユーザ数
        likes: “50”
        clips: “50”
        follows: “50”
        comments: “50”
        posts: “50”
        favtags: “50”
    }]
}
```

図 D.4: API の仕様書 4

◆transition/registrationのJSONフォーマット

```
{
	start: 2014-01-01, // 開始日
	end: 2015-01-01, // 終了日
	data : [{
		date: “2015-04-01”, // 日付
		registrations: “100” // 登録人数
	}]
}
```

==

Actions の定義

VIEW	写真詳細またはタイムラインを見る
LIKE	「いいね！」を押下
CLIP	写真をクリップ
FOLLOW	他ユーザをフォロー
COMMENT	写真にコメント
POST	写真を投稿
FAVTAG	タグをお気に入りに追加（モバイルのみ）

※ 空集合は**NOVIEW**（写真詳細またはタイムラインが見られていない）とする
※ flow/allのみ，**NEW**（新規登録）ノードが存在する

==

NodeIDとの対応関係

以下を参照
https://docs.google.com/spreadsheets/d/1T3nR2XL1Y5iunix2WHj0OuXpO2sdeh3Tchk2FWBPP4w/edit?usp=sharing

図 D.5: API の仕様書 5

付 録E　ノードID対応表一覧

65 コのノードの ID とアクションの対応表を示す。

nodeid	VIEW	LIKE	CLIP	FOLLOW	COMMENT	POST	FAVTAG
1	1	0	0	0	0	0	0
2	1	1	0	0	0	0	0
3	1	0	1	0	0	0	0
4	1	1	1	0	0	0	0
5	1	0	0	1	0	0	0
6	1	1	0	1	0	0	0
7	1	0	1	1	0	0	0
8	1	1	1	1	0	0	0
9	1	0	0	0	1	0	0
10	1	1	0	0	1	0	0
11	1	0	1	0	1	0	0
12	1	1	1	0	1	0	0
13	1	0	0	1	1	0	0
14	1	1	0	1	1	0	0
15	1	0	1	1	1	0	0
16	1	1	1	1	1	0	0
17	1	0	0	0	0	1	0
18	1	1	0	0	0	1	0
19	1	0	1	0	0	1	0
20	1	1	1	0	0	1	0
21	1	0	0	1	0	1	0
22	1	1	0	1	0	1	0
23	1	0	1	1	0	1	0
24	1	1	1	1	0	1	0
25	1	0	0	0	1	1	0
26	1	1	0	0	1	1	0
27	1	0	1	0	1	1	0
28	1	1	1	0	1	1	0
29	1	0	0	1	1	1	0
30	1	1	0	1	1	1	0

図 E.1: ノード ID 対応表 1

31	1	0	1	1	1	1	0
32	1	1	1	1	1	1	0
33	1	0	0	0	0	0	1
34	1	1	0	0	0	0	1
35	1	0	1	0	0	0	1
36	1	1	1	0	0	0	1
37	1	0	0	1	0	0	1
38	1	1	0	1	0	0	1
39	1	0	1	1	0	0	1
40	1	1	1	1	0	0	1
41	1	0	0	0	1	0	1
42	1	1	0	0	1	0	1
43	1	0	1	0	1	0	1
44	1	1	1	0	1	0	1
45	1	0	0	1	1	0	1
46	1	1	0	1	1	0	1
47	1	0	1	1	1	0	1
48	1	1	1	1	1	0	1
49	1	0	0	0	0	1	1
50	1	1	0	0	0	1	1
51	1	0	1	0	0	1	1
52	1	1	1	0	0	1	1
53	1	0	0	1	0	1	1
54	1	1	0	1	0	1	1
55	1	0	1	1	0	1	1
56	1	1	1	1	0	1	1
57	1	0	0	0	1	1	1
58	1	1	0	0	1	1	1
59	1	0	1	0	1	1	1
60	1	1	1	0	1	1	1
61	1	0	0	1	1	1	1

図 E.2: ノード ID 対応表 2

62	1	1	0	1	1	1	1
63	1	0	1	1	1	1	1
64	1	1	1	1	1	1	1
65	0	0	0	0	0	0	0
66	新規登録ユーザ（flow/allのみ使用）						

図 E.3: ノード ID 対応表 3

付 録F　テーブル定義書

Redshift および RDS に実装したテーブルの定義書を示す。

テーブル情報

テーブル型	ファクトテーブル	**作成者**	河越 嵩介	
論理テーブル名	ユーザアクティビ	**作成日**	2015/08/06	
物理テーブル名	user_activity	**更新日**	2015/10/08	
スキーマ	public			
【備考】				
その日（date）に行われた各ユーザの能動的な行動の合計数を格納す				

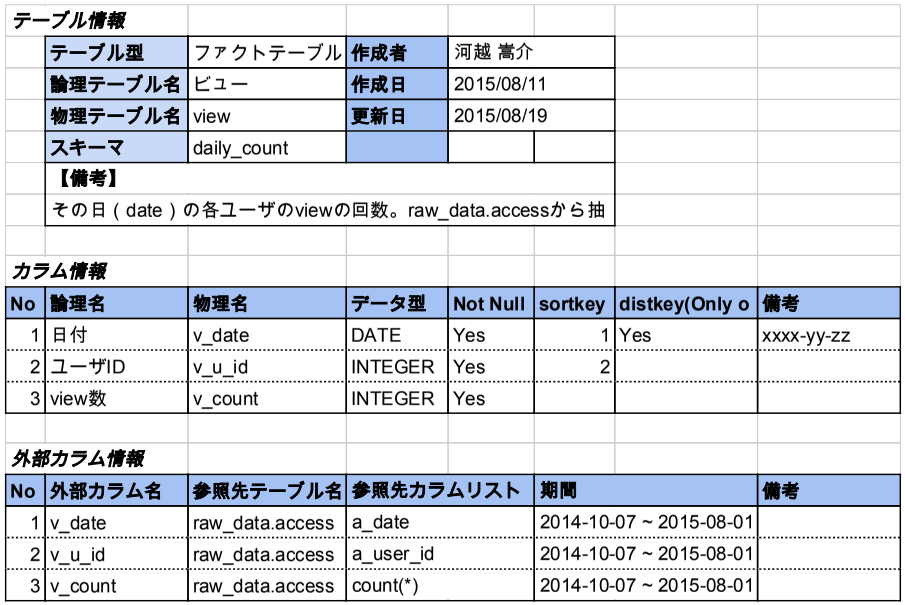
カラム情報

No	論理名	物理名	データ型	Not Null	sortkey	distkey(Only one	備考
1	日付	u_date	DATE	Yes	1	Yes	xxxx-yy-zz
2	ユーザID	u_id	INTEGER	Yes	2		
3	view数	u_view	SMALLINT	Yes			これが1以上、つまり一回でも画面を見たユーザのアクティビティだけがこのテーブルに格納されている
4	like数	u_like	SMALLINT	Yes			
5	clip数	u_clip	SMALLINT	Yes			
6	follow数	u_follow	SMALLINT	Yes			
7	comment数	u_comment	SMALLINT	Yes			
8	post数	u_post	SMALLINT	Yes			
9	favtag数	u_favtag	SMALLINT	Yes			2.0から追加

外部カラム情報

No	外部カラム名	参照先テーブル名	参照先カラムリスト	期間	備考
3	u_view	daily_count.view	v_count	2014-10-08 ~ 2015-08-01	viewはaccessを元に作成
4	u_like	daily_count.like	l_count	2012-04-12 ~ 2015-05-21	likeはphoto_moreを元に作成
5	u_clip	daily_count.clip	cl_count	2013-12-05 ~ 2015-05-21	clipはphoto_clipsを元に作成
6	u_follow	daily_count.follow	f_count	2012-04-09 ~ 2015-05-21	followはuser_followを元に作成
7	u_comment	daily_count.comme	cm_count	2012-04-12 ~ 2015-05-21	commentはphoto_commentを元に作成
8	u_post	daily_count.post	ps_count	2012-04-12 ~ 2015-05-21	postはphotosを元に作成
9	u_favtag	daily_count.fav_tag	ft_count	2012-04-09 ~ 2015-05-21	favtagはfav_tag（request）を元に作成

図 F.1: user_activity



テーブル情報

テーブル型	ファクトテーブル	**作成者**	河越 嵩介		
論理テーブル名	ビュー	**作成日**	2015/08/11		
物理テーブル名	view	**更新日**	2015/08/19		
スキーマ	daily_count				
【備考】					
その日（date）の各ユーザのviewの回数。raw_data.accessから抽					

カラム情報

No	論理名	物理名	データ型	Not Null	sortkey	distkey(Only o	備考
1	日付	v_date	DATE	Yes	1	Yes	xxxx-yy-zz
2	ユーザID	v_u_id	INTEGER	Yes	2		
3	view数	v_count	INTEGER	Yes			

外部カラム情報

No	外部カラム名	参照先テーブル名	参照先カラムリスト	期間	備考
1	v_date	raw_data.access	a_date	2014-10-07 ~ 2015-08-01	
2	v_u_id	raw_data.access	a_user_id	2014-10-07 ~ 2015-08-01	
3	v_count	raw_data.access	count(*)	2014-10-07 ~ 2015-08-01	

図 F.2: view

テーブル情報

テーブル型	ファクトテーブル	**作成者**	河越 嵩介	
論理テーブル名	ライク	**作成日**	2015/08/18	
物理テーブル名	like	**更新日**	2015/08/18	
スキーマ	daily_count			
【備考】				
その日（date）の各ユーザのlikeの回数。rwa_data.likeから抽出				

カラム情報

No	論理名	物理名	データ型	Not Null	sortkey	distkey(Only o	備考
1	日付	l_date	DATE	Yes	1	Yes	xxxx-yy-zz
2	ユーザID	l_u_id	INTEGER	Yes	2		
3	like数	l_count	INTEGER	Yes			

外部カラム情報

No	外部カラム名	参照先テーブル名	参照先カラムリスト	期間	備考
1	l_date	raw_data.like	l_date	2012-04-12 ~ 2015-05-21	
2	l_u_id	raw_data.like	l_u_id	2012-04-12 ~ 2015-05-21	
3	l_count	raw_data.like	count(*)	2012-04-12 ~ 2015-05-21	

図 F.3: like

テーブル情報

テーブル型	ファクトテーブル	**作成者**	河越 嵩介		
論理テーブル名	クリップ	**作成日**	2015/08/11		
物理テーブル名	clip	**更新日**	2015/08/18		
スキーマ	daily_count				
【備考】					
その日（date）の各ユーザのclipの回数。rwa_data.clipから抽出					

カラム情報

No	論理名	物理名	データ型	Not Null	sortkey	distkey(Only o	備考
1	日付	cl_date	DATE	Yes	1	Yes	xxxx-yy-zz
2	ユーザID	cl_u_id	INTEGER	Yes	2		
3	clip数	cl_count	INTEGER	Yes			

外部カラム情報

No	外部カラム名	参照先テーブル名	参照先カラムリスト	期間	備考
1	cl_date	raw_data.clip	cl_date	2013-12-05 ~ 2015-05-21	
2	cl_u_id	raw_data.clip	cl_u_id	2013-12-05 ~ 2015-05-21	
3	cl_count	raw_data.clip	count(*)	2013-12-05 ~ 2015-05-21	

図 F.4: clip

テーブル情報

テーブル型	ファクトテーブル	**作成者**	河越 嵩介		
論理テーブル名	フォロー	**作成日**	2015/08/18		
物理テーブル名	follow	**更新日**	2015/08/18		
スキーマ	daily_count				
【備考】					
その日（date）の各ユーザのfollowの回数。rwa_data.followから抽					

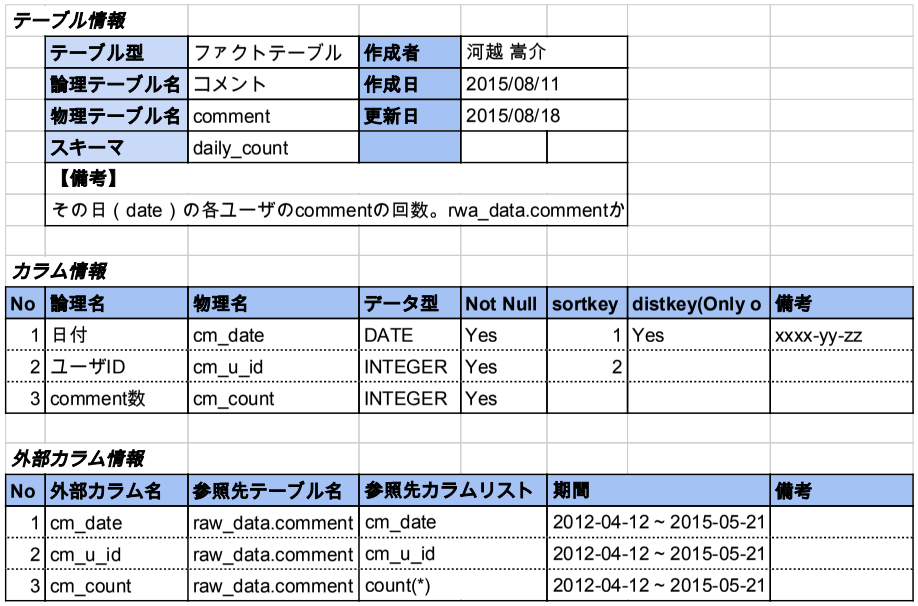
カラム情報

No	論理名	物理名	データ型	Not Null	sortkey	distkey(Only o	備考
1	日付	f_date	DATE	Yes	1	Yes	xxxx-yy-zz
2	ユーザID	f_u_id	INTEGER	Yes	2		
3	follow数	f_count	INTEGER	Yes			

外部カラム情報

No	外部カラム名	参照先テーブル名	参照先カラムリスト	期間	備考
1	f_date	raw_data.follow	f_date	2012-04-09 ~ 2015-05-21	
2	f_u_id	raw_data.follow	following_id	2012-04-09 ~ 2015-05-21	
3	f_count	raw_data.follow	count(*)	2012-04-09 ~ 2015-05-21	

図 F.5: follow



テーブル情報

テーブル型	ファクトテーブル	**作成者**	河越 嵩介	
論理テーブル名	コメント	**作成日**	2015/08/11	
物理テーブル名	comment	**更新日**	2015/08/18	
スキーマ	daily_count			
【備考】				
その日（date）の各ユーザのcommentの回数。rwa_data.commentか				

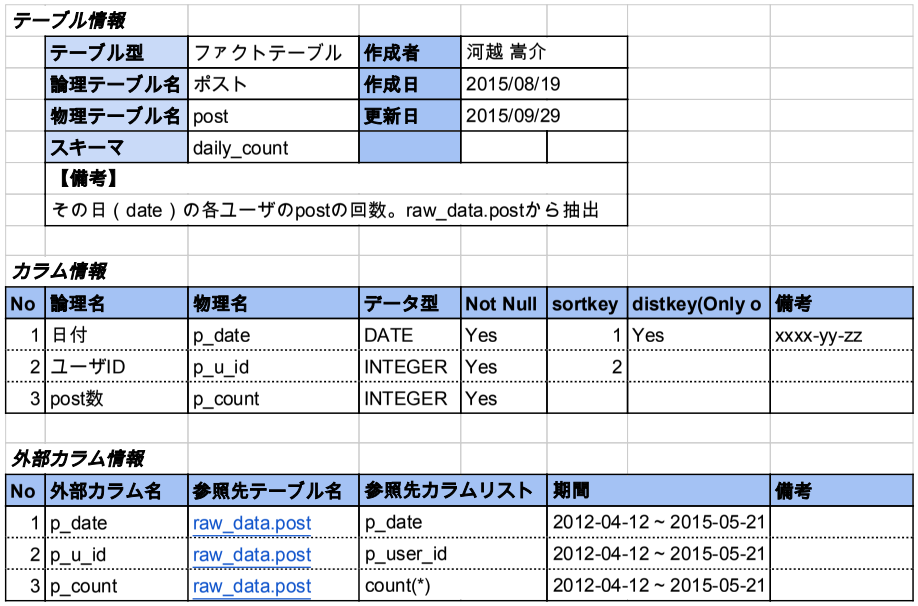
カラム情報

No	論理名	物理名	データ型	Not Null	sortkey	distkey(Only o	備考
1	日付	cm_date	DATE	Yes	1	Yes	xxxx-yy-zz
2	ユーザID	cm_u_id	INTEGER	Yes	2		
3	comment数	cm_count	INTEGER	Yes			

外部カラム情報

No	外部カラム名	参照先テーブル名	参照先カラムリスト	期間	備考
1	cm_date	raw_data.comment	cm_date	2012-04-12 ~ 2015-05-21	
2	cm_u_id	raw_data.comment	cm_u_id	2012-04-12 ~ 2015-05-21	
3	cm_count	raw_data.comment	count(*)	2012-04-12 ~ 2015-05-21	

図 F.6: comment



テーブル情報

テーブル型	ファクトテーブル	**作成者**	河越 嵩介	
論理テーブル名	ポスト	**作成日**	2015/08/19	
物理テーブル名	post	**更新日**	2015/09/29	
スキーマ	daily_count			
【備考】				
その日（date）の各ユーザのpostの回数。raw_data.postから抽出				

カラム情報

No	論理名	物理名	データ型	Not Null	sortkey	distkey(Only o	備考
1	日付	p_date	DATE	Yes	1	Yes	xxxx-yy-zz
2	ユーザID	p_u_id	INTEGER	Yes	2		
3	post数	p_count	INTEGER	Yes			

外部カラム情報

No	外部カラム名	参照先テーブル名	参照先カラムリスト	期間	備考
1	p_date	raw_data.post	p_date	2012-04-12 ~ 2015-05-21	
2	p_u_id	raw_data.post	p_user_id	2012-04-12 ~ 2015-05-21	
3	p_count	raw_data.post	count(*)	2012-04-12 ~ 2015-05-21	

図 F.7: post

テーブル情報

テーブル型	ファクトテーブ	**作成者**	河越 嵩介	
論理テーブル名	ファボタグ	**作成日**	2015/10/08	
物理テーブル名	fav_tag	**更新日**	2015/10/08	
スキーマ	daily_count			
【備考】				
その日（date）の各ユーザのviewの回数。raw_data.fav_tagから抽出				

カラム情報

No	論理名	物理名	データ型	Not Null	sortkey	distkey(Only one)	備考
1	日付	ft_date	DATE	Yes	1	Yes	xxxx-yy-zz
2	ユーザID	ft_u_id	INTEGER	Yes	2		
3	view数	ft_count	INTEGER	Yes			

外部カラム情報

No	外部カラム名	参照先テーブル	参照先カラムリスト	期間	備考
1	ft_date	raw_data.fav_tag	ft_date	2012-04-09 ~ 2015-05-21	
2	ft_u_id	raw_data.fav_tag	ft_user_id	2012-04-09 ~ 2015-05-21	
3	ft_count	raw_data.fav_tag	count(*)	2012-04-09 ~ 2015-05-21	

図 F.8: fav_tag

テーブル情報							
	テーブル型	ディメンションテー	**作成者**	河越 嵩介			
	論理テーブル名	ノード	**作成日**	2015/08/06			
	物理テーブル名	node	**更新日**	2015/11/04			
	スキーマ	public					
	【備考】						
	nodeidの対応表						
カラム情報							
No	**論理名**	**物理名**	**データ型**	**Not Null**	**sortkey**	**distkey(Only o**	**備考**
1	nodeid	n_id	SMALLINT	Yes	1		
2	view	n_view	SMALLINT	Yes			
3	like	n_like	SMALLINT	Yes			
4	clip	n_clip	SMALLINT	Yes			
5	follow	n_follow	SMALLINT	Yes			
6	comment	n_comment	SMALLINT	Yes			
7	post	n_post	SMALLINT	Yes			
8	favtag	n_favtag	SMALLINT	Yes			Web版にはない

図 F.9: node

テーブル情報

テーブル型	ファクトテーブル	**作成者**	河越 嵩介
論理テーブル名	トレーニングデー	**作成日**	2015/10/23
物理テーブル名	td_ua	**更新日**	2015/12/07
スキーマ	public		
【備考】			
機械学習のための学習データ。user_activityにcount_daysが追加されている			

カラム情報

No	論理名	物理名	データ型	Not Null	sortkey	distkey(Only one)	備考
1	view数	view	SMALLINT	Yes			これが1以上、つまり一回でも画面を見たユーザのアクティビティだけがこのテーブルに格納されている
2	like数	likee	SMALLINT	Yes			
3	clip数	clip	SMALLINT	Yes			
4	follow数	follow	SMALLINT	Yes			
5	comment数	comment	SMALLINT	Yes			
6	post数	post	SMALLINT	Yes			
7	favtag数	favtag	SMALLINT	Yes			
8	7日の間にアクセスした日数	count_days	SMALLINT	Yes			

図 F.10: td_ua

付 録G　因子分析のプロット一覧

7.4 で行った因子分析の結果をプロットしたものを示す。

G.1　スクリープロット

G.2　因子負荷量

G.3　因子間の相関

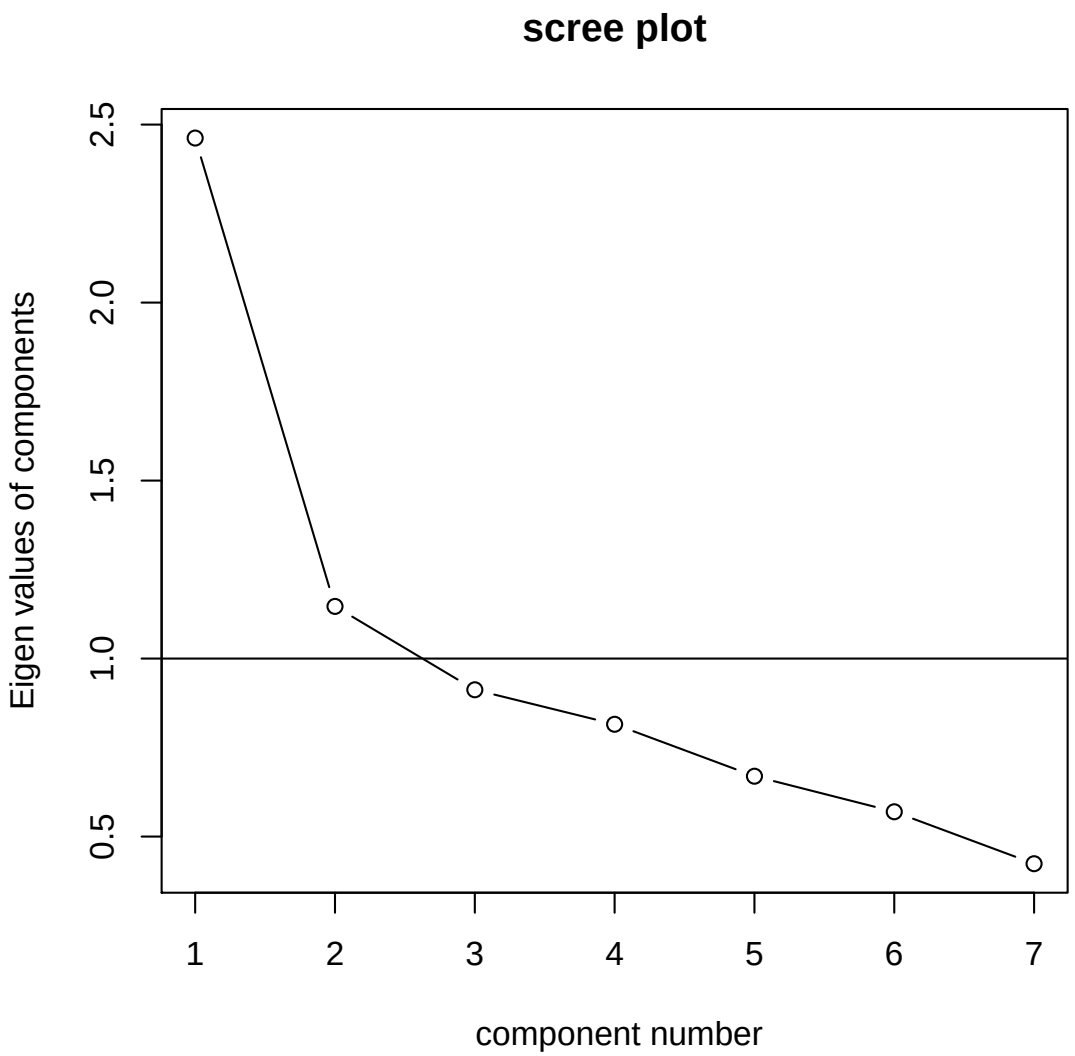


図 G.1: スクリープロット

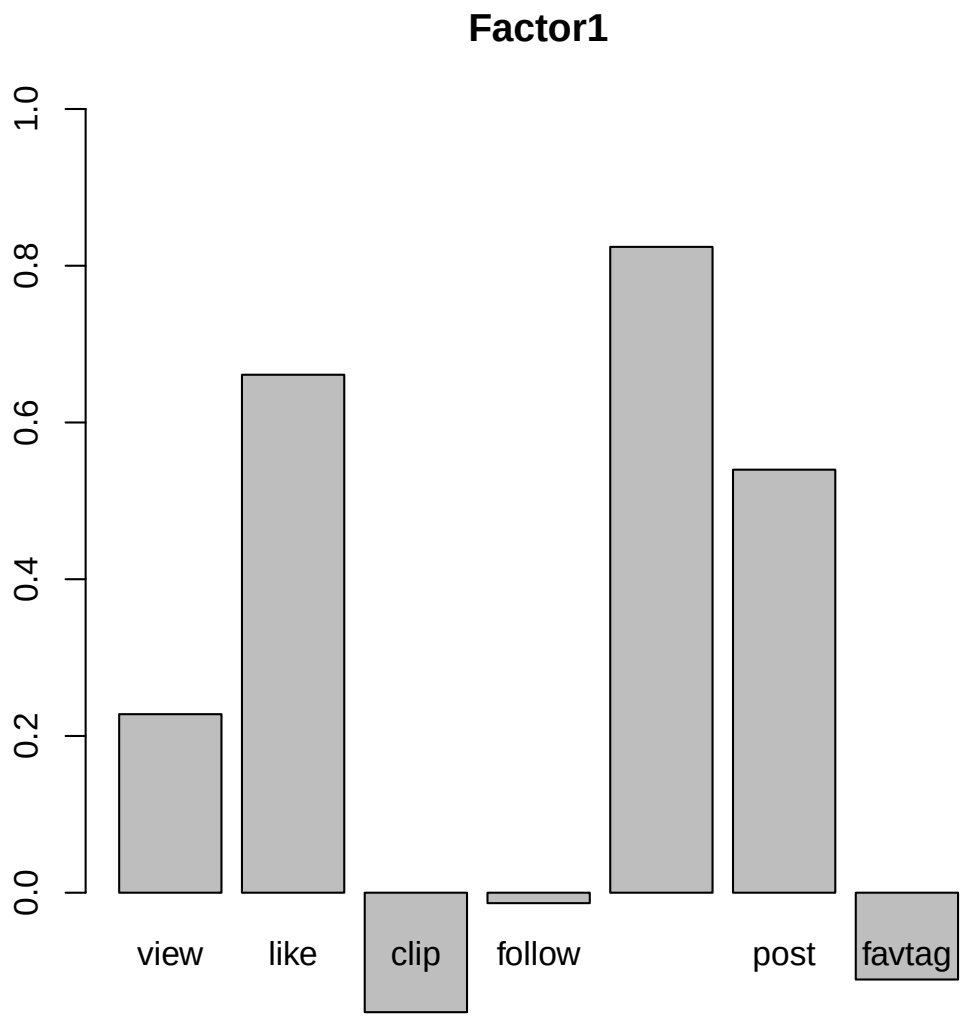


図 G.2: 第 1 因子の負荷量

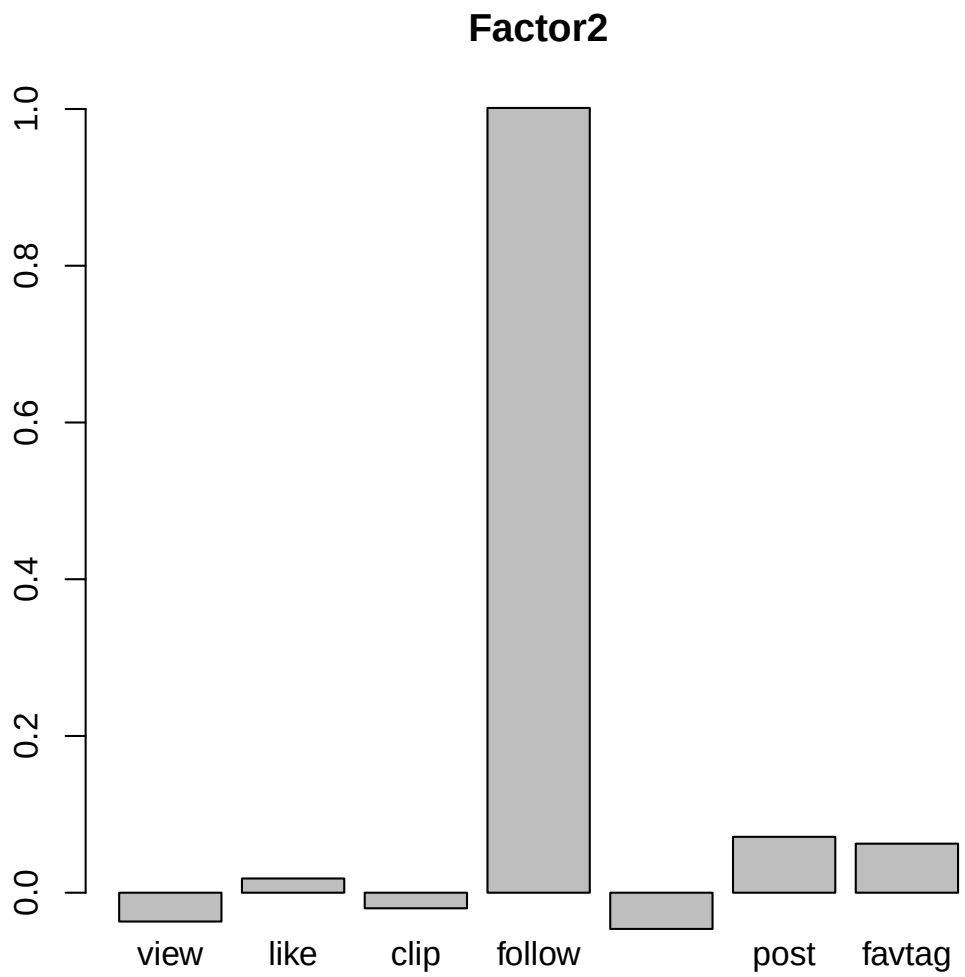


図 G.3: 第 2 因子の負荷量

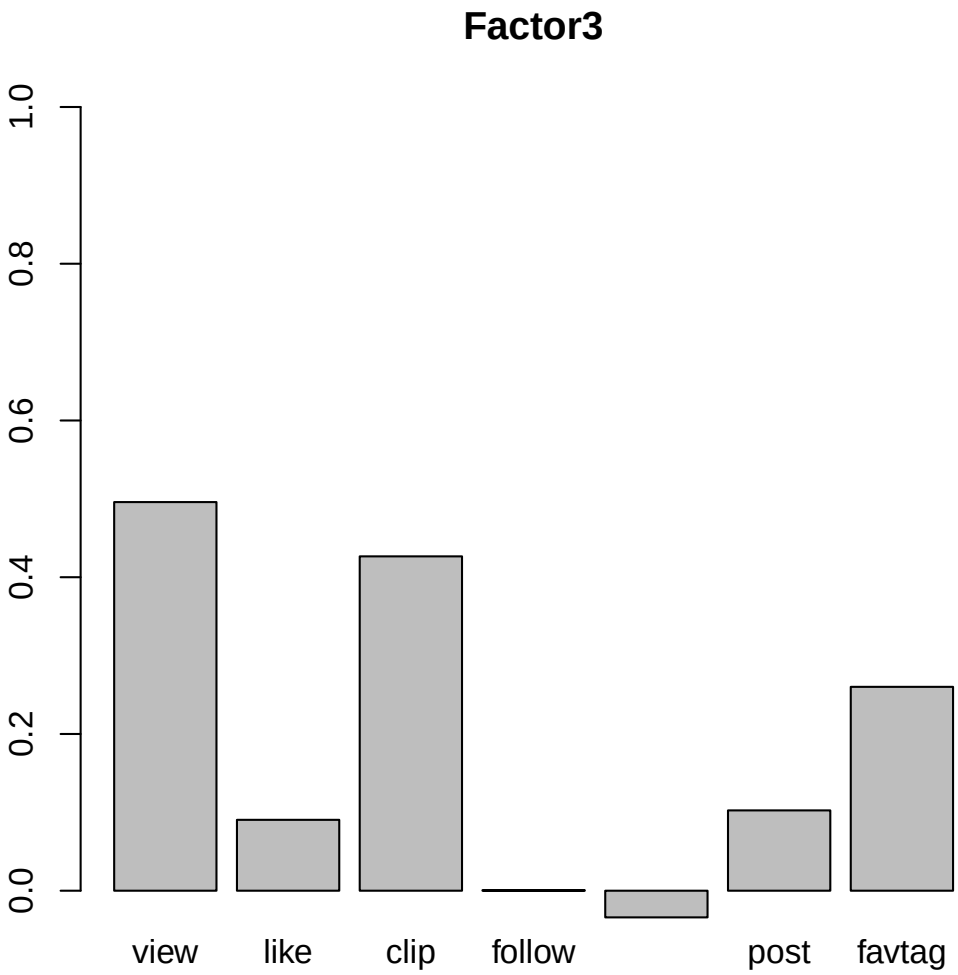


図 G.4: 第 3 因子の負荷量

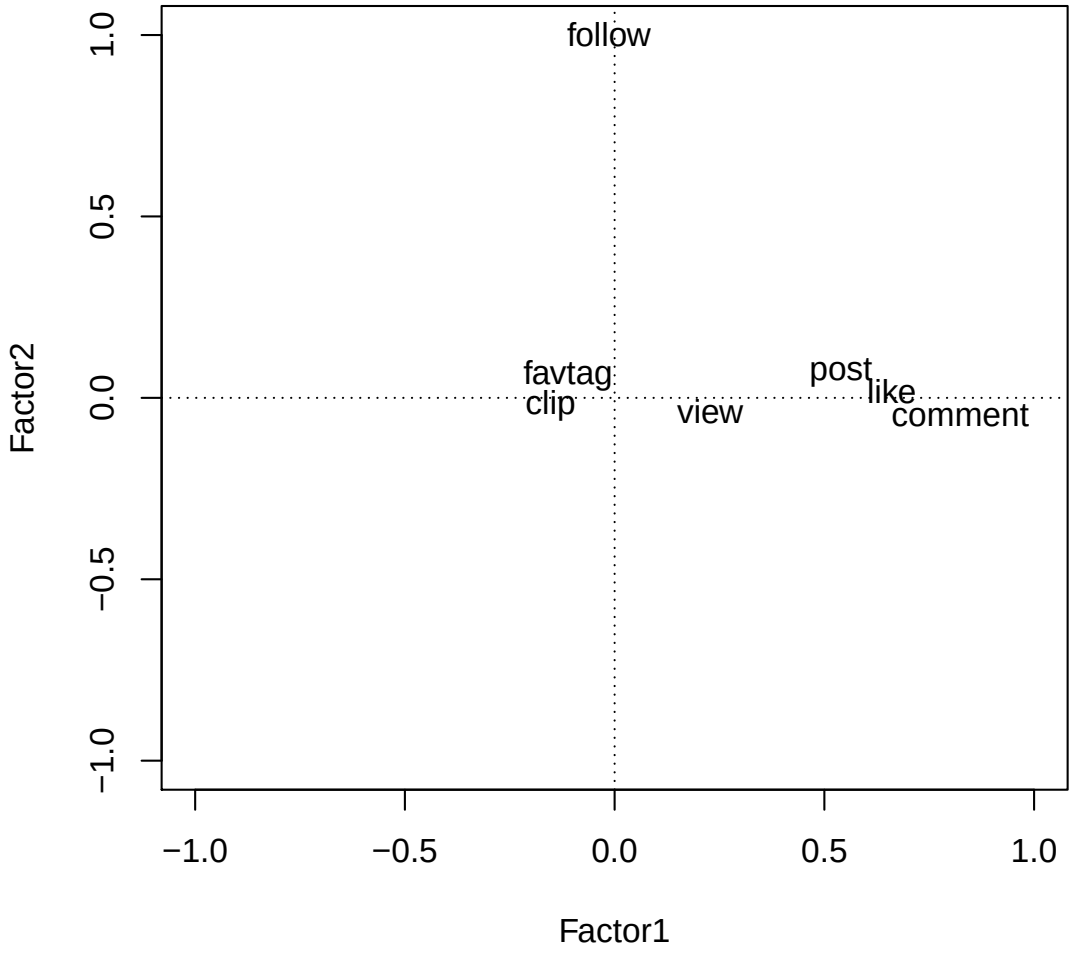


図 G.5: 第 1 因子と第 2 因子の相関

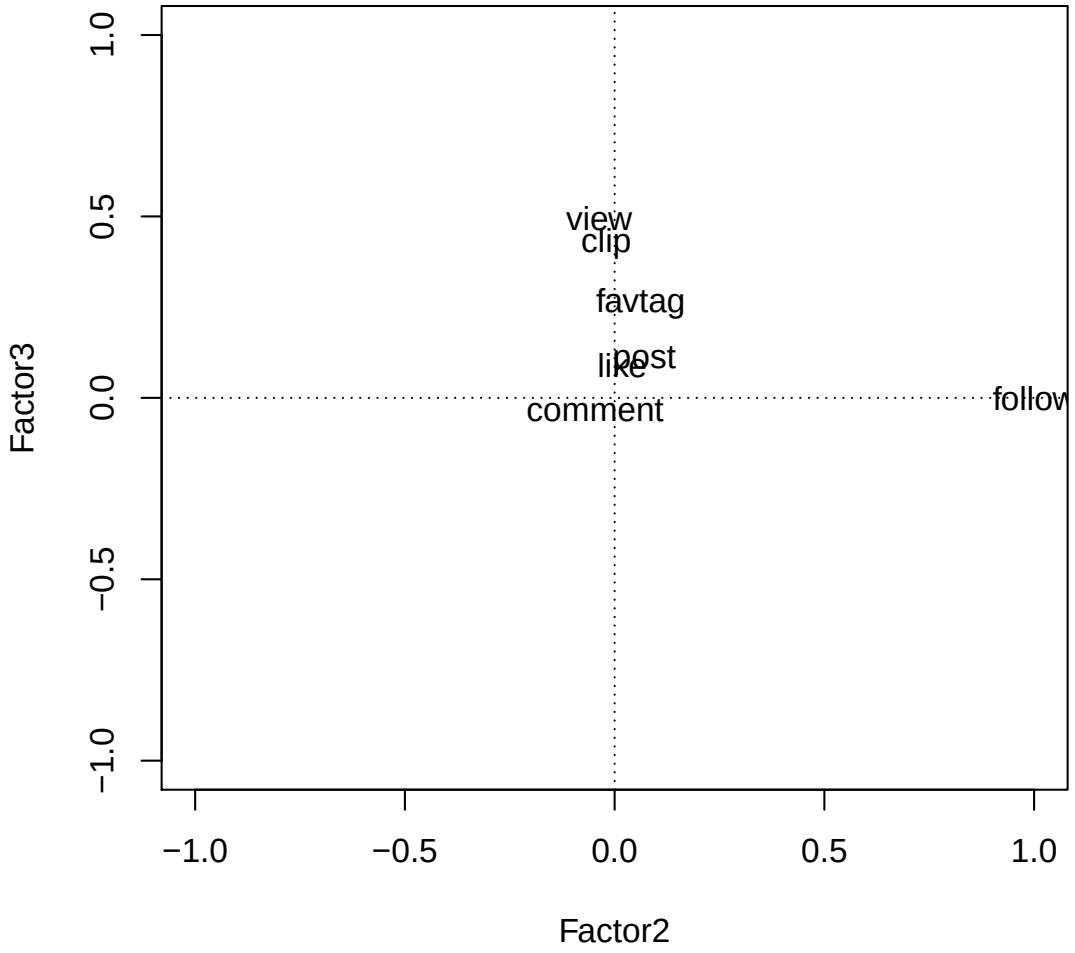


図 G.6: 第 2 因子と第 3 因子の相関

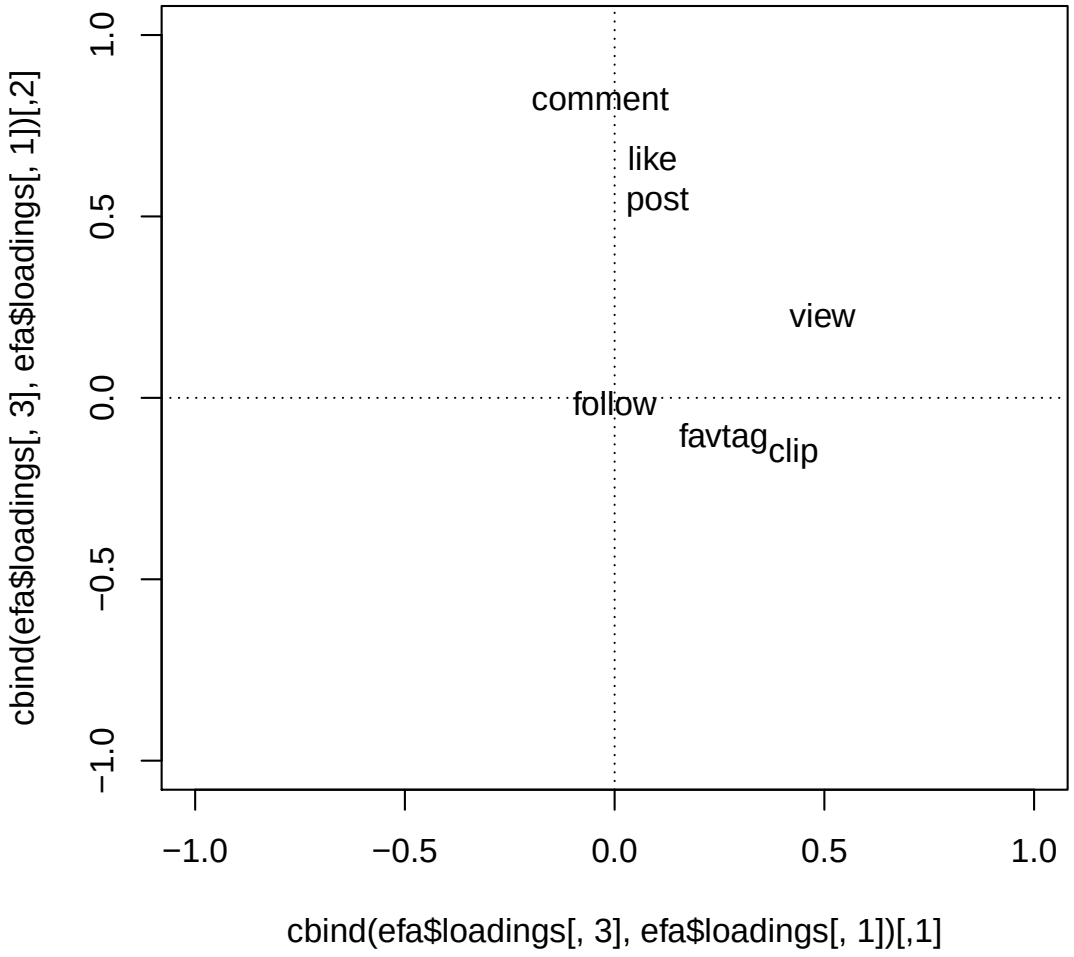


図 G.7: 第 3 因子と第 1 因子の相関