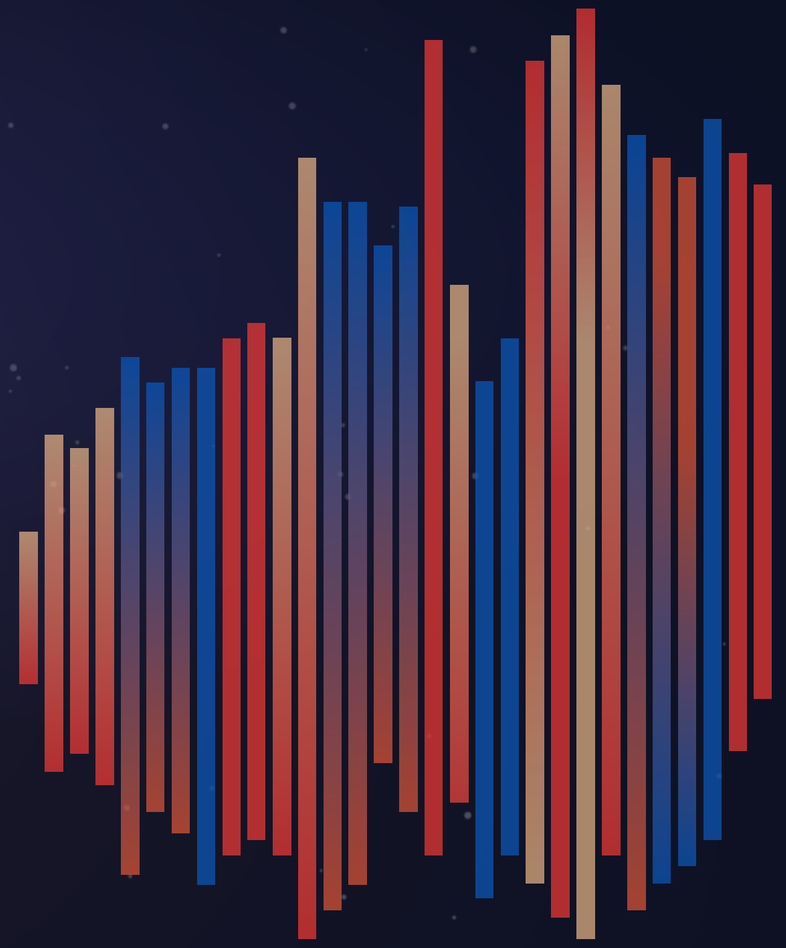
Game analytics 100: The retention curve

by Russell Ovans, East Side Games

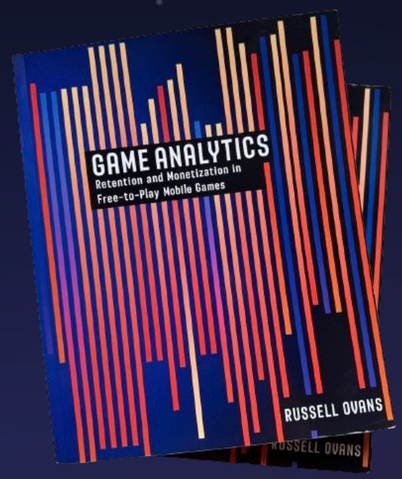
April 2024



$DAU_n = r(n) \times \text{cohort size}$ $PD_n = \sum_{i=0}^{n} DiR = \sum_{i=0}^{n} r(i)$ $LTV_n = ARPDAU \times \sum_{i=0}^{n} r(i)$ $DnR = \frac{DAU_n}{\text{installs}}$ $r(n) = an^b$

Table of contents

Portions of this paper previously appeared in the book Game Analytics: Retention and Monetization in Free-to-Play Games. Reprinted with permission of Thought Pilots.



Introduction

Retaining customers is the life blood of any business, let alone a game studio. Generating new customers is an expensive endeavour as it requires outlays of cash on advertising. As such, it is generally more economical to keep the customers you have than it is to buy new ones. Or, as my Uncle Bob explained to me years ago, “Once the customer enters your business, all of your effort shifts to selling them one thing: a return visit.”

The topic of this paper is mobile game customer retention: how do we quantify and model the rate at which users return to play our games? You can only monetize the users you have, so if your game doesn’t retain its players, you will struggle to generate revenue. Most game analysts are surely familiar with the notion of day-*n* retention; i.e., the proportion of your players who return to play exactly *n* days after they install and play their first session. For example, if 100 users install your game on July 1st, and 20 of those players return to play on July 8th, then day 7 retention (D7R) is 0.20. In this paper, we generalize the concept of day-*n* retention with a **retention curve**, a simple formula to model and predict player retention for any day after install. We describe how a retention curve is derived from a set of historical data, plus two important applications of the curve: predicting daily active users (DAU) from a constant number of daily installs; and, the summation of this curve is the expected number of distinct days a new user will play your game.

Preliminary definitions

Data analysts tend not to think too much about individual players. Instead, descriptive statistics are drawn from a group of players called a cohort. A **cohort** is a set of players who have something in common. Normally, this is their install date, but additional attributes can be used to determine membership in a cohort. For example, a cohort might consist of all the Android players from the US who installed version 1.26.5 on July 1, 2023. Cohorts define the players used in the calculation of averages and other descriptive statistics that make up key performance indicators (KPIs) such as *day-n* retention.

Retention is a KPI that is modified with a "days since install" index, which we denote with the variable *n*. You never talk about retention without specifying a particular day after a cohort's install date, which is indicated by "D*n*." **D*n* retention** (D*n*R) is the proportion – often expressed as a percentage – of a cohort that plays exactly *n* days after their install date: not the day before, nor the day after. By definition, D0R is always 1.0 since the install date is equivalent to the date of a player's first session. D1R is the proportion of installs who played at any time in the calendar date immediately following their install date. The higher your retention, the better. For the idle-genre titles published by East Side Games, a good D1R is around 40%. For D7R, we aim for 20%.

D*n* retention as a KPI is measured at a standard set of days since install, typically for $n \in \{1, 7, 30, 90\}$. But in general, D*n*R can be measured for any day since install as

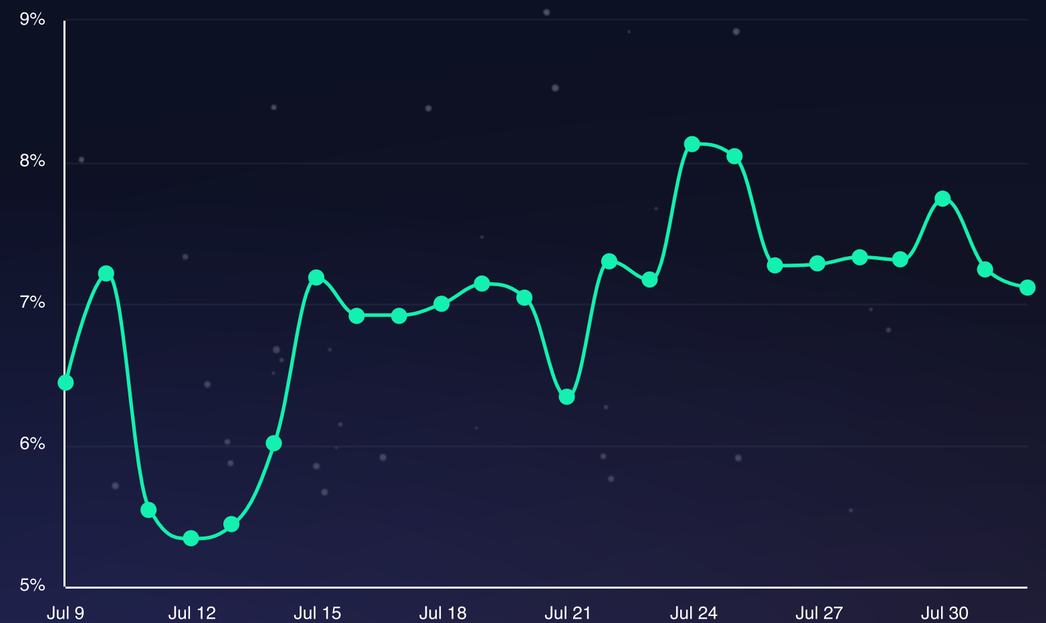
$$\mathrm{D}n\mathrm{R} = \frac{\mathrm{DAU}_n}{\mathrm{installs}}$$

where *installs* is the size of the cohort, and DAU_n is a count of the daily active users from the cohort who played on the n^{th} day after their install date. Note that $\mathrm{DAU}_0 \equiv \mathit{installs}$.

For a cohort to have a value for any D*n* retention metric their install date must be more than *n* days ago. For example, we can't calculate D7R for a cohort of installs until eight days after their install date, at which point we say the cohort and its users are "seven days **fully baked**." If a cohort is not fully baked with respect to a KPI, the value is null.

A **time series** is made up of successive measurements – over

consistent intervals – of the same variable. When charted or graphed, time is the independent variable and occupies the *x*-axis. The *x*-axis usually refers to a specific date when a game was played; e.g., DAU by day. But for other time-series KPIs, the *x*-axis can represent a cohort *install* date that measures the evolving, future behaviour of only those users who installed the game on that date. Day-*n* retention is one such cohort-based time series. If we generate a time-series graph of D30 retention, the value on July 1st represents the proportion of the installs from that date who played on July 31st. The value on July 2nd represents the proportion who installed the game that date and played on August 1st, and so on. (The presented data is representative, and not from any particular game.)



Retention is an aggregate measure applied to a group of players. But at the individual user level, D*n* retention is simply a binary variable: the user either played on that day (indicated by the value 1), they did not play (indicated by a 0), or if not yet fully baked, we don't know yet (indicated by a null value). Here is sample retention data for some random players.

user_id	install_date	d1	d7	d30	d90
858993	2017-04-23	0	1	0	0
2630131	2017-06-28	1	0	0	1
8554854	2021-11-19	0	0	0	0
226962	2017-04-20	1	1	1	0
7706121	2021-02-06	0	0	0	0
9025650	2023-09-28	0	0	0	0
3221035	2017-08-19	1	1	0	0

Rather than a time series of daily retention values, a **retention profile** for a game as a whole is constructed by taking a weighted average of D*n* retention values over multiple install cohorts. Given a database table similar to the one above, this is trivial to compute with a SQL query, the result set of which might appear as follows:

d1r	d7r	d30r	d90r
0.3712	0.1626	0.0787	0.0394

Typically, users are taken from a range of consecutive install dates (as specified by a WHERE clause in the SQL) that are at least 90 days old to ensure only fully-baked cohorts are included, but if not, the nulls are conveniently skipped over and not included in the calculation of the average.

How GameAnalytics displays retention

In GameAnalytics, retention metrics are displayed using cohorts, which are groups of players who share common attributes, such as install date or specific in-game actions. The retention metrics, such as day-*n* retention (D*n*R), are calculated based on the behavior of these cohorts over time. By default, GameAnalytics allows users to view retention metrics via default triggers, such as the first session after install.

GameAnalytics Pro users have additional options, including the ability to define custom start trigger event conditions and custom return trigger event conditions. This allows for more granular analysis and customization of retention metrics based on specific player actions or behaviors.

Additionally, GameAnalytics provides the possibility to apply global filters to easily alter static dimensions like country, enabling users to analyze retention metrics across different player segments.

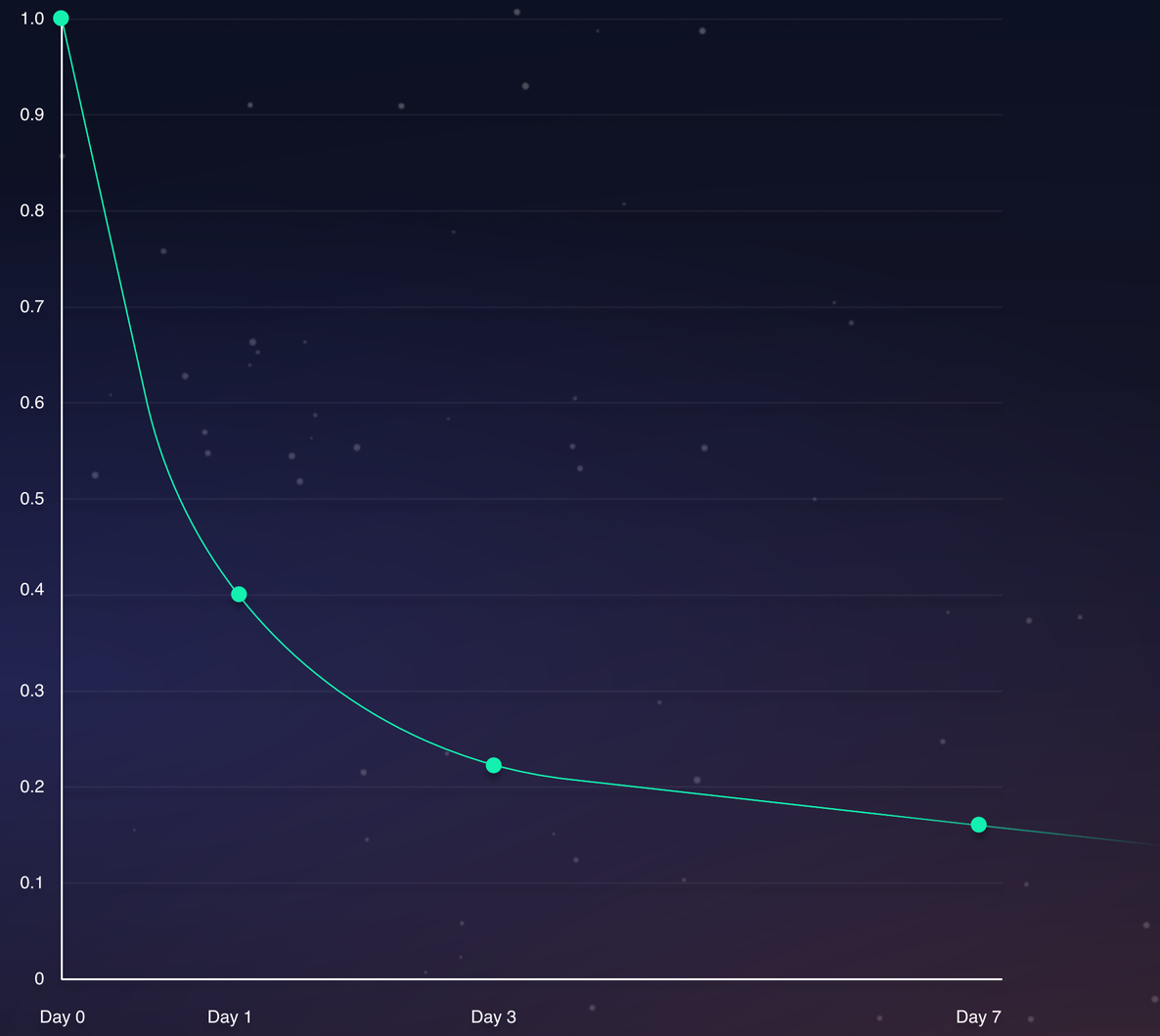
By utilizing GameAnalytics, developers can gain insights into player retention patterns, identify areas for improvement, and optimize their games to enhance player engagement and satisfaction.

The retention curve

By convention, the days 1, 7, 30, and 90 after install are included in a game's core set of KPIs. But what about all the other days in between or after these specific days? We could measure every DnR from the cohort data, but is there a closed form function that can tell us with reasonable accuracy what D*n* retention is for any day *n?* It turns out there is, and this function defines the retention curve. The **retention curve** is a mechanism to fully describe a historical retention profile and predict the expected number of days each new user will engage with your game before churning.

Retention curves are built from a retention profile of observed D*n* values. Each value is a dot, and the retention curve connects the dots. Here is a curve (the green line) fit to observed retention values of 0.4, 0.23, and 0.16 at D1, D3, and D7, respectively.

Unlike the time-series graph introduced previously, the *x*-axis is *n*, representing a discrete day since install. Note how the curve predicts retention values well beyond the data points used in its construction. The curves that best fit mobile game retention



profiles tend to be real-valued power functions. The generic form of a retention power curve *r* is a function r: $\mathbb{N} \rightarrow [0, 1]$ of days since install *n*:

$$r(n)=an^b$$

where $a \in (0, 1]$, $b \in [-1, 0)$, $r(0)=1.0$.

The parameters *a* and *b* are referred to as the coefficient and exponent, respectively. The coefficient's value mimics D1R. The exponent is negative, which means that retention starts out high but decays over time. Values returned by the function are proportions between 0 and 1. For example, a retention curve for a mobile game might be:

$$r(n)=0.4n^{-0.5}$$

This specific function evaluates to a D1 retention of 40.0%, D7R of 15.12%, and D180R of 2.98%. But it can also calculate retention for any arbitrary day after install; e.g., D53R is 5.49%.

Fitting a retention curve $\boldsymbol{r(n)=an^b}$ to a set of D*n*R observations is an exercise in statistical regression: determine values for *a* and *b* that minimize the residual error between the observations (the actual data) and the predictions (the values returned by the function). We now consider two tools a data analyst might employ in performing this regression: *Excel* and *Tableau*.

Fitting a retention curve in Excel

Assume we have the following observed retention profile for a game in soft launch, which we have entered into an Excel spreadsheet. The observations are for D1, D3, and D7 retention.

	A	B
1	n	DnR
2	0	1
3	1	0.4
4	3	0.23
5	7	0.16

What is a power function $\boldsymbol{r(n)=an^b}$ that best describes these data points? In Excel, we can deploy the LINEST function to fit a line or curve to an array of known *y*- and *x*-values, where *y* is the

dependent variable and *x* is the independent variable. In our case, *n* (the days since install) is the independent variable. As its name suggests, the LINEST function – by default – fits a line and returns an array (two adjacent cells) containing the slope and *y*-intercept of the formula that best fits the data. To instead fit a power function to our data, we need to pass LINEST the logarithm of the array of known values. Because the logarithm of 0 is undefined, we only include three data points:

```
=LINEST(LN(B3:B5),LN(A3:A5))
```

	A	B	C
1	n	DnR	
2	0	1	
3	1	0.4	
4	3	0.23	
5	7	0.16	
6	b	ln_a	a
7	-0.472	-0.927	0.396

If we enter that function into cell A7, Excel returns the exponent *b* and the natural log of the coefficient *a* in the cells A7 and B7, respectively. Cell C7 contains the function =EXP(B7) in order to convert the coefficient to the correct form.

Our retention curve is thus:

$$r(n)=0.396n^{-0.472}$$

And that is how you use Excel to determine a formula for a game's retention curve based on a small sample of D*n*R values. Congratulations – achievement unlocked!

This retention curve – based on observed D1, D3, and D7 retention metrics only – predicts the following values for long-tail retention:

Day *n*	Predicted D*n* Retention
14	0.114
30	0.079
60	0.057
90	0.047
180	0.034
360	0.025
720	0.018

1.8% D720 retention? Really? When modeled as a power function, there is no terminal day beyond which every player is guaranteed to have churned, i.e., $r(n)>0$ for all $n \in \mathbb{N}$. Depending on your game's live operations, elder-game mechanics, and content release schedule, this may or may not be a reasonable assumption.

Constructing a retention curve in Tableau from (more) historical data

How well can we trust these estimates for D90+ retention? They seem optimistic, perhaps owing to the very early and limited number of data points. With more and later observations of retention, the regression analysis should become more accurate. Let's consider a mature game that launched 90 days ago and start by capturing retention data for each user's first 30 days. We include only those users who had a chance to play 30 days after install (i.e., those who installed between 90- and 31-days prior) and simply count the number of those users who played n days after their install date.

days_since_install	players	installs	retention
0	7891	7891	1.0
1	2929	7891	0.3712
2	2120	7891	0.2687
3	1769	7891	0.2242
4	1559	7891	0.1976
5	1409	7891	0.1786
6	1326	7891	0.1680
7	1283	7891	0.1626
...	...	...	...
28	658	7891	0.0834
29	651	7891	0.0825
30	621	7891	0.0787

Notice we do not cohort by install date – all we care about is how many users played exactly n days after they installed, regardless of each user's specific install date.

The first row *(days_since_install = 0)* is the total number of users who installed between 90- and 31-days prior. In this case, 7,891 installs are included in our analysis; this is the size of the entire cohort spanning multiple install dates. To calculate retention as a proportion of cohort size, we divide the players by the installs. For example, we see that 1,283 played seven days after they installed: D7 retention is 1283/7891= 6.26%.

Utilizing Tableau to explore this dataset, we plot *retention* as a function of *days_since_install (see the top chart)*.

Note: Some might object to the use of a line chart instead of a bar chart given that the x-axis is discrete and not continuous; i.e., we don't deal in fractional days since install. My use of a continuous line is not meant to imply that something is happening between samples; it is simply an aid to help visualize a trend in the data. Besides, Tableau won't overlay a trend line on a bar chart.

Now from the Analytics tab we add a Trend Line of type Power to fit a curve to the data *(see the bottom chart)*.

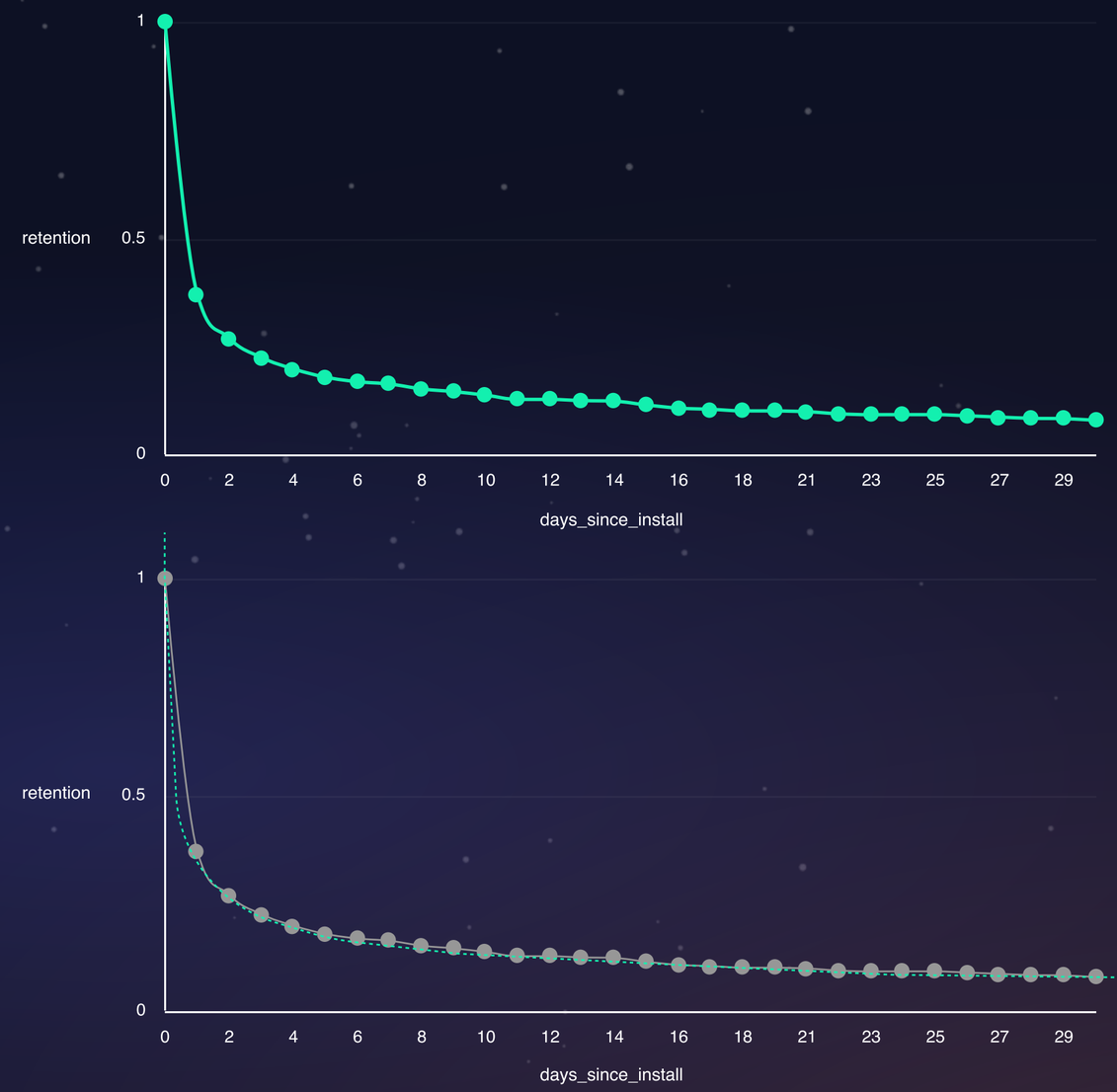
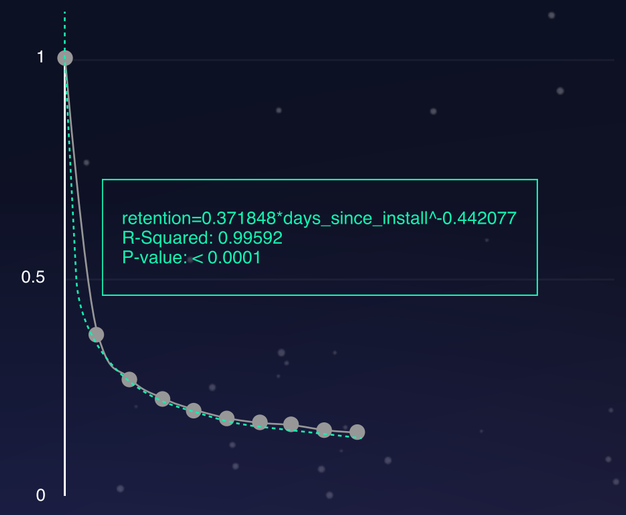


Tableau uses the same method of least-squares as Excel to find values for the coefficient *a* and exponent *b* that best fits these observations. If we hover over the dashed line, a pop-up indicates the formula that Tableau has decided is the best fit for our data.

To three decimal places of accuracy, we have:

$$r(n)=0.372n^{-0.442}$$

This curve predicts D90R of 5.09% and D180R of 3.75%. We'll take that!



Predicting DAU with a retention curve

By now we should all be very comfortable with the idea that a retention curve is a model – derived from historical data – defined by a power function $r(n)=an^b$. This function defines the probability a player has a session exactly *n* days after their install date. Therefore, when applied to a cohort of installs, the expected number of DAU from that cohort on day *n* after install is:

$$DAU_n = r(n) * cohort\ size$$

Calculating the steady-state daily active users given a retention formula and a constant number of daily new installs can be accomplished with lots of copying and pasting in a spreadsheet, but that approach is tedious and error prone. For example, assume a new game launches with a retention curve defined by $0.4n^{-0.5}$ and 100 installs per day. After seven days, a spreadsheet can tell us the expected total DAU from the overlapping cohorts is 260. See the table, where each column represents a single install cohort, and the last column is the total DAU by day *n* after launch.

n									DAU
0	100								**100**
1	40	100							**140**
2	28	40	100						**168**
3	23	28	40	100					**191**
4	20	23	28	40	100				**211**
5	18	20	23	28	40	100			**229**
6	16	18	20	23	28	40	100		**245**
7	15	16	18	20	23	28	40	100	**260**

By examination of this table, we notice a pattern: the DAU by day *n* is a function of only the original cohort's retained users and the DAU on the prior day. For example, DAU after seven days is 260, comprised of 15 players still remaining from the very first cohort, plus the 245 DAU from the prior day. This pattern is succinctly expressed as a recurrence relationship:

$$DAU_0=100$$
$$DAU_n=(r[n]*DAU_0)+DAU_{n-1}$$

What is this sorcery! I'm not enough of a mathematician to come up with a closed form solution to that recurrence relation, but as a recovering computer scientist I know how to convert it to a recursive function. Here it is in R:

```
DAU_n <- function(n)
{
  if (n == 0) {
    return (100)
  }
  return (round(100 * 0.4 * n ^ -0.5) + DAU_n(n-1))
}

R> for (i in 0:7) print(c(i, DAU_n(n=i)))
[1]   0 100
[1]   1 140
[1]   2 168
[1]   3 191
[1]   4 211
[1]   5 229
[1]   6 245
[1]   7 260
```

Goodbye spreadsheet! It's a straightforward exercise to generalize this function to work for any retention curve – a power function parameterized by coefficient *a* and exponent *b* – and any number of installs per day. First we define two functions: one to generate a list of a retention curve's daily values, and another to generate a list of the running sum of these daily values. Both functions are recursive and work backwards from *n* to 0.

```
ret_curve <- function(a, b, n)
{
  if (n <= 0) return (1)
  return(c(ret_curve(a, b, n-1), a * n^b))
}

sum_ret_curve <- function(a, b, n)
{
  if (n >= 0)
     ret_curve(a, b, n) + c(0, sum_ret_curve(a, b, n-1))
}
```

Here are the functions in action, revealing **r(n)** and **∑ r(n)** for the first *n=7* days after install for a retention curve defined by ***a=0.4*** and ***b=-0.5***.

```
R> options(digits=4)
R> ret_curve(a=0.4, b=-0.5, n=7)

[1] 1.0000 0.4000 0.2828 0.2309 0.2000 0.1789 0.1633 0.1512

R> sum_ret_curve(a=0.4, b=-0.5, n=7)

[1] 1.000 1.400 1.683 1.914 2.114 2.293 2.456 2.607
```

Finally, we can compute a list of expected DAU from game launch to *n* days after, assuming *installs*/day and a retention curve defined by an^b:

```
dau <- function(installs, a, b, n)
{
  floor(installs * sum_ret_curve(a, b, n))
}
```

For example, the predicted DAU after 30 days for a new game with 500 installs per day and a retention profile of **a=0.372**, **b=-0.440** is 2,510:

```
R> dau(installs=500, a=0.372, b=-0.442, n=30)

  [1]  500  686  822  937 1038 1129 1213 1292 1366 1437 1504
1568 1630 1690 1748

 [16] 1804 1859 1912 1964 2014 2064 2112 2160 2206 2252 2297
2341 2384 2427 2469 2510
```

Because the retention curve asymptotically approaches 0 as $n \to \infty$, this DAU model grows forever. A simple fix is to define a terminal date after which all players are assumed churned, then update the function `ret_curve` accordingly.

Player duration as the summation of the retention curve

The retention curve defines the probability that a user drawn at random plays exactly *n* days after their install date. As such, an important use of the retention curve is in predicting future engagement of new and existing players. In particular, the number of distinct dates we can expect each new user to play the game during their first *n* days after install.

A **play date** occurs when a user has at least one session on a specific date. The **player duration (PD)** is the count of a user's distinct play dates from install. Let PDn be the average player duration within n days of install for a cohort of users. Then

$$PDn = \sum_{i=0}^{n} DiR = \sum_{i=0}^{n} r(i)$$

As a cohort metric, PD*n* is a random variable with an expected value defined by the summation of the retention curve ***r(n)***. Once you have a function *r* that estimates the retention profile of your game, you can use this curve to predict the player duration of new installs, and lifetime value (LTV) if you multiply by average revenue per daily active user (ARPDAU). For example, what is the average player duration during the first 30 days after install for a cohort whose retention curve is defined by $r(n)=0.372n-0.442$?

We reuse one of our R functions and determine this value is approximately five:

```
R> sum_ret_curve(a=0.372, b=-0.442, n=30)

[1] 1.000 1.372 1.646 1.875 2.076 2.259 2.427 2.585 2.733
2.874 3.009 3.137 3.261 3.381 3.497 3.609 3.719 3.825 3.929
[20] 4.030 4.129 4.226 4.321 4.414 4.505 4.595 4.683 4.769
4.855 4.939 5.021
```

Note: In the literature you will often see reference to the "area under the curve", or "the integral of the retention curve." Strictly speaking this is incorrect, as the retention function is only defined for integer values of n. If you plug a retention curve formula into a symbolic integration tool, you will find the answer is typically inflated compared to a discrete summation.

These five play dates can occur anywhere within a user's first 30

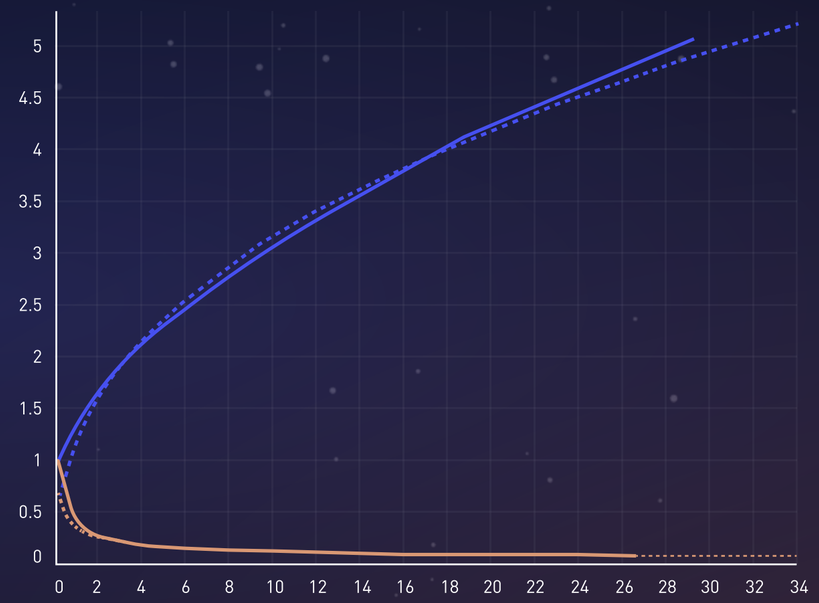
and are not necessarily consecutive. Keep in mind that this is a statistical mean and by no means reflects the median number of days one can expect a random user to play during the first 30 days after install. Most installs only play one or two days before churning, but this mean is skewed by a few regular users who like the game and play nearly every day.

Why is PD*n* important? Well, if we know that ARPDAU is $1.00, we can assume that LTV30 for this cohort is likely to be 5.021 * $1.00=$5.02. In practice, when launching a new version of a game, extrapolating a retention curve ***r(n)*** from its early D*n*R signals and taking an average ARPDAU from existing players, then a reasonable estimate of LTV is given by

$$LTV_n = ARPDAU * \sum_{i=0}^{n} r(i)$$

Rather than model the retention curve and perform a summation, the expected value of PD*n* can be modelled by fitting a curve directly to the running sum of observed daily retention values. In other words, instead of building a retention curve from the daily retention rates, we fit a curve to the running sum of these same retention rates. The resulting closed form function **PD(*n*)** can estimate expected player duration for any value of *n* without requiring the calculation of a summation.

To illustrate, refer to the following chart where both the retention (in orange) and sum of retention (in blue) are plotted on the same synchronized axes:



The blue curve is implemented in Tableau with a Table Calculation that builds a summation over the individual retention metrics that make up the orange curve. At day zero, we have 100% retention and 1.00 player day. By day 30, the player days are 5.017. (This is the sum of the actual D*n*R observations, whereas 5.021 is the sum of the retention curve that estimates this data.) The dashed lines are Trend Lines of type Power, resulting in the following closed form estimate of PD*n*:

$$PD(n)=1.21n^{0.41}$$

We can use this formula to extrapolate player duration to D90 (=7.65) and D180 (=10.17). If average ARPDAU is $1.00, then LTV90 and LTV180 are estimated to be $7.65 and $10.17, respectively.

Retention benchmarks

GameAnalytics offers users access to comprehensive Retention benchmarks, providing valuable insights into player engagement and retention performance across the gaming industry. These benchmarks serve as a reference point for developers to compare their game's retention metrics against industry standards and identify areas for improvement.

These benchmarks are derived from GameAnalytics dataset compiled from various games and developers worldwide, offering a comprehensive overview of retention trends across different genres, platforms, and player demographics.

One of the key features of Retention benchmarks in GameAnalytics is the ability to customize and filter the data based on specific criteria. Users can adjust filters such as genre, platform, region, and player demographics to refine their benchmark comparisons and identify relevant insights for their game.

Additionally, GameAnalytics provides Engagement, Monetization, and Advertising benchmarks.

*While paying attention to your game performance cannot be overstated, it is equally important to understand player preferences and industry standards. Conveniently packaged in **GameAnalytics Pro**, our industry benchmarks can help you uncover players' behavior patterns alongside the access to next-gen analytics solutions for your game. Learn more here.*

Summary

Retaining existing users is fundamental to the success of any mobile game since you can only monetize the players you have. Retention is measured as the proportion of a cohort that plays exactly n days after their install date:

$$DnR = \frac{DAU_n}{DAU_0}$$

...where DAU_0 is the size of the cohort, and DAU_n is a count of the daily active users from the cohort who played on the n^{th} day after their install date.

The retention profile for a game is a weighted average of the observed D*n*R values for installs over a range of historical dates, typically calculated for days 1, 7, 30, and 90 since install.

A non-linear regression function fit to the retention profile succinctly captures expected player interaction for all past and future installs. This function is referred to as the retention curve. For mobile games, the retention curve is typically a negative-exponent power function $r(n)=an^b$, which defines a proportion as a function of *n*, the days since install. It follows that $DnR \cong r(n)$. Each game will have its own values for *a* and *b* that best fit their retention profile.

Since it defines the probability that a user plays exactly *n* days after install, expected DAU by day *n* after game launch is dependent on the retention curve. The recurrence relationship is $DAU_n=(r[n]*DAU_0)+DAU_{n-1}$, which can be implemented as a recursive function in any programming language.

Player duration (PD) is the number of distinct dates a new user plays over their lifetime (i.e., until churn) or within their first *n* days after install. The expected value of PD by day *n* is the summation of the retention curve from days 0 to *n*. This summation can be performed both iteratively with a programming language or estimated analytically by fitting a curve *PD(n)* to the running sum of *r(n)*. LTV*n* is predicted by multiplying *PD(n)* by ARPDAU.

About the book

Portions of this paper previously appeared in the book Game Analytics: Retention and Monetization in Free-to-Play Games. Reprinted with permission of Thought Pilots.

Mobile games are big business, and the landscape is more competitive than ever. With an in-depth focus on the core areas of user retention and predicting customer lifetime value, *Game Analytics* contains the hands-on SQL queries, R scripts, statistical theory, full-colour Tableau visualizations, and insider tips and tricks you need to succeed as a data analyst, product manager, or user acquisition manager in free-to-play games.

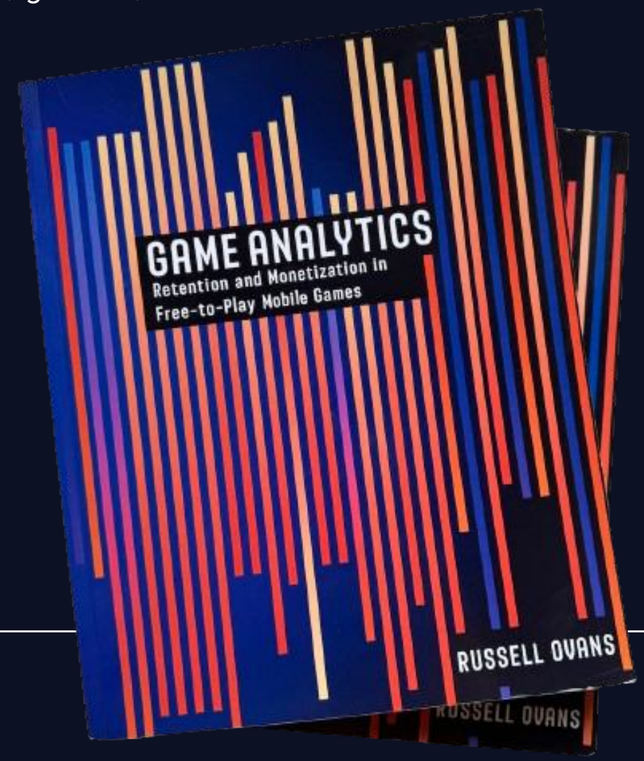
Game Analytics describes in detail how successful game studios make money, collect and query player data, define key performance indicators (KPIs), build dashboards and predictive models of retention and monetization, measure and predict return on ad spend (ROAS), and use statistics to analyze A/B tests designed to improve retention and monetization.

The book is available on Amazon in various countries.

About the author

Russell Ovans, Ph.D., was the Director of Analytics at East Side Games, developers of hit mobile games such as *The Office: Somehow We Manage* and *Trailer Park Boys: Greasy Money*. He is a computer scientist and has worked as both a software engineering professor and programmer for over 35 years. In 2007, he founded Backstage Technologies, a social game studio that pioneered the monetization of free-to-play games on Facebook. Best known for its *Family Feud* app, Backstage was acquired by RealNetworks in 2010, after which Russ returned to teaching college, worked as an executive-in-residence at a tech incubator, and opened a brewery. He returned to the games industry in 2018 to lead analytics, growth, and ad monetization at ESG, a tenure during which the company quadrupled revenue and went public.

He welcomes your feedback: *russell.ovans@gmail.com*.



GameAnalytics