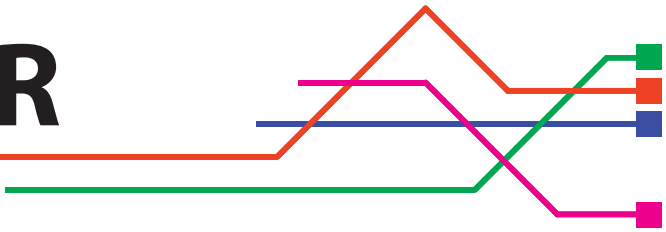
SLOT DESIGNER



Tools for professional mathematicians

Elements Of Slot Design

3rd Edition

GameDesignAutomation.com

http://slotdesigner.com/

Document: Elements of Slot Design 009.docx

CHAPTER 1 INTRODUCTION

This document is intended as an introduction to Slot game mathematics and design. In time it should expand to provide a comprehensive tutorial and reference to all aspects of Slot design.

The fundamentals of Slot mathematics are introduced so that games can be designed from the ground up in Microsoft Excel, code, or using Slot Designer. It assumes a basic understanding of probability and Slot machines.

There is no standardized method of performing these calculations, designing spreadsheets, or even terminology. Different countries and companies for example refer to symbols as icons, and reel strips as reel bands. This document therefore uses the terminology defined by Slot Designer, which is the commercial product of the author.

While I had intended to produce a very much larger book, 9 years have gone by since the last edition and somehow it didn't happen. I've been busy working on Slot Designer.

This update coincides with the release of the next generation of Slot Designer. Some of the added material is of a more philosophical nature; I've learned a lot and had time to more deeply reflect of the nature of what it is I'm doing with slot design tools and I'd like to share.

I appreciate any feedback as to the content and future direction, and my thanks to those who have done so over the years.

Robert Muir

muir@GameDesignAutomation.com

CHAPTER 2 THE BASICS

We'll start by considering a very simple slot game with 3 reels, each of 10 symbols, 1 payline, and 1 pay rule, and a cost to play each game of 1 credit. Each reel contains 10 symbols in total, with 1 A and 9 X symbols,

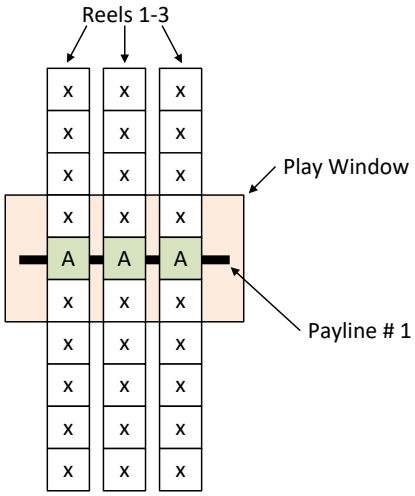


Figure 1. Simple 3x3 Game

The game's single pay rule awards 1000 credits when we line up 3 A symbols across the payline. When spinning the reels we can see the probability of stopping with an A on the payline for each reel is 1/10, hence the probability of getting 3 A's across the 3 reels is 1/1000.

$$p(A,A,A) = \frac{1}{10} \times \frac{1}{10} \times \frac{1}{10} = \frac{1}{1000}$$ Equation 1

When we win the rule awards 1000 credits, so the average win per game is 1 credit

$$average\,win = prize \times probability\ of\ \ prize$$ Equation 2

$$average\,win = prize \times p(a,a,a) = 1000 \times \frac{1}{1000} = 1\,credit$$

When we play the game (spin the reels) we pay 1 credit, so the average cost to play each game (the bet) is simply 1 credit.

The RTP (return to player) of a game is defined as the average win as a percentage of the bet, which for this game is 100%.

$$RTP = \frac{average\,win}{average\,bet} \times 100\%$$ Equation 3

$$RTP = \frac{1}{1} \times 100\% = 100\%$$ Equation 4

In other words, over the long term as we play this game, we will win 100% of what we bet even though we win only once in 1000 games. When playing a game with an RTP over 100% the player wins money over the long term, and below 100% the player loses money.

If we were to change the prize value from 1000 credits to 500 credits, then the average win (Equation 2) is 0.5 credits, and the RTP (Equation 3) is 50%.

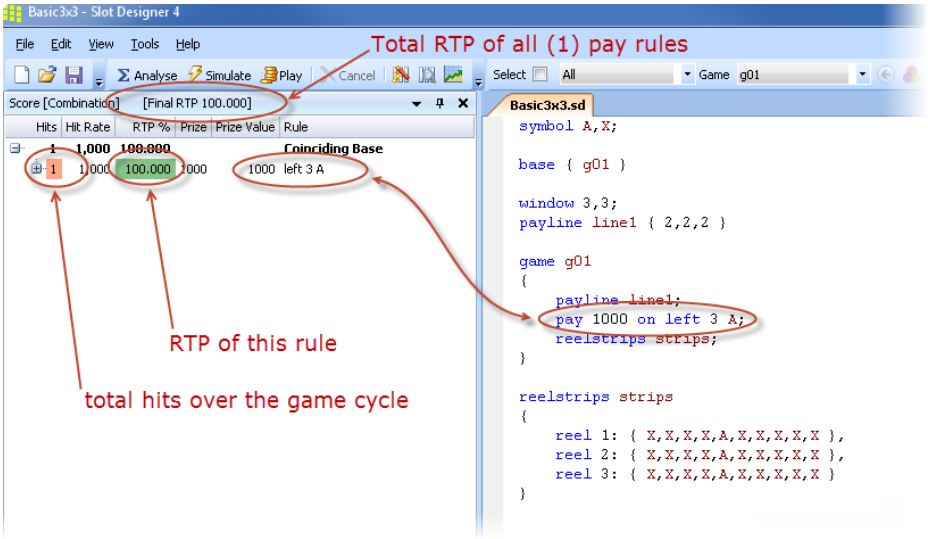
If we were to add a second A symbol on the first reel only (while keeping the length of 10 symbols), we would double the probability of A on that reel, hence doubling the probability of 3 A occurring on the payline. With the original 1000 credit prize, the average win is then 2 credits (Equation 2), and the RTP 200% (Equation 3). We now win once in every 500 games.

$$\begin{aligned} p(A,A,A) &= \frac{2}{10} \times \frac{1}{10} \times \frac{1}{10} \\ &= \frac{2}{1000} \\ &= \frac{1}{500} \end{aligned}$$

Combining a second A on the first reel with the prize value of 500 credits, we're doubling the probability and halving the prize, hence the RTP remains at 100%. With 2 A on each of the 3 reels, we would have 8 times the probability, resulting in an RTP of 400% (with the 500 credit prize).

The winning pattern of 3 A symbols is called a "combination", and the complete list of combinations comprises the game's pay table. We can write this as {A,A,A}.

In Slot Designer we can implement the game with the script shown on the right, and the basic statistical report for the pay rules on the left.



COUNTING HITS

Rather than perform all the calculations as probabilities, we usually count the number of winning combinations over the game cycle and divide by the cycle at the end. The difference is subtle and end result is the same, but it's easier to work with. We can rewrite Equation 1 using hits instead of probability

$$\begin{aligned} p(A,A,A) &= \frac{hits}{cycle} \\ &= \frac{1 \times 1 \times 1}{10 \times 10 \times 10} \\ &= \frac{1}{1000} \end{aligned}$$

Equation 5

HIT RATE AND HIT FREQUENCY

The hit rate tells us how many games we have to play on average to win a particular prize, while the hit frequency tells us what percentage of games win the prize. This is particularly useful when looking at how often the player will trigger a feature game.

The hit rate can be calculated several different ways

$$hitrate = \frac{1}{probability} \quad \text{Equation 6}$$

$$hitrate = \frac{cycle}{hits} \quad \text{Equation 7}$$

And the hit frequency

$$hit\ \ frequency = \frac{100\%}{hit\ rate} \quad \text{Equation 8}$$

In the game of Equation 1 the hit rate is 1000, or 1000 games between hits.

$$\begin{aligned} hit\ rate(A,A,A) &= \frac{cycle}{hits} \\ &= \frac{1000}{1} \\ &= 1000 \end{aligned} \quad \text{Equation 9}$$

And the hit frequency is 0.1%

$$\begin{aligned} hit\ \ frequency(A,A,A) &= \frac{100\%}{1000} \\ &= 0.1\% \end{aligned} \quad \text{Equation 10}$$

VOLATILITY

If we have a game that wins 1 credit for every play, the average win is then 1 credit, and with an average bet of 1 credit this gives us an RTP of 100% (Equation 3). In the previous section we also calculated a RTP of 100% for a game that wins 1000 credits once in every 1000 games, and zero every other play.

These two games have the same RTP, yet would be fundamentally different to play. The key difference between these two games lies in their volatility. The game that wins every time has a very low volatility, and the other a very high volatility. Indeed, their volatility is so extreme that neither would be a practical design.

We can also see that as the chance of winning increases the prize won must correspondingly decrease to achieve the same RTP.

SIMPLE PAY RULES

We can add a second pay rule to the game in Figure 1, which wins 100 credits when we get A on the left 2 reels, but not on the 3rd reel. This is commonly referred to as a left to right combination, and we can write it as the combinations {A,A,#A}, where # means "not".

Counting the hits for this rule we have 1 on reels 1 and 2 as before, but on reel 3 there are 9 hits where A does not occur. We don't count 10 hits for 'any' symbol as matching the combination on reel 3, as another A on reel 3 would match the left 3 A pay rule instead. Left 2 A and left 3 A are mutually exclusive.

The number of hits is then

$$\begin{aligned} hits(left\ 2\ A) &= hits(A, A, \#A) \\ &= 1 \times 1 \times (10-1) \\ &= 9 \end{aligned}$$ Equation 11

The average win due to this pay rule is then 0.9 credits (using Equation 2 and Equation 5)

$$\begin{aligned} average\ win &= prize \times \frac{hits(left\ 2\ A)}{cycle} \\ &= 100 \times \frac{9}{1000} \\ &= 0.9\ credits \end{aligned}$$ Equation 12

Hence, using Equation 3, the RTP contributed to the game by this pay rule is 90%. The total RTP of the game is the sum of the individual contributions, and since the left 3 A rule contributes 100% the total RTP is therefore 190%.

Instead of a left to right pay rule we could add instead an 'any' pay rule, such that any 2 A pays 100 credits. This pay rule comprises 3 combinations, {A,A,#A}, {A,#A,A}, and {#A,A,A}, each of which has exactly 2 A symbols, and are mutually exclusive. There are 27 hits on any 2 A

$$\begin{aligned} hits(any\ 2\ A) &= hits(A, A, \#A) + hits(A, \#A, A) + hits(\#A, A, A) \\ &= 1 \times 1 \times (10-1) + 1 \times (10-1) \times 1 + (10-1) \times 1 \times 1 \\ &= 9 + 9 + 9 \\ &= 27 \end{aligned}$$ Equation 13

And the RTP for this pay rule is therefore 270%, and the total RTP for the game is 370%.

$$\begin{aligned} RTP(any\ 2\ A) &= prize \times \frac{hits(any\ 2\ A)}{cycle \cdot bet} \\ &= 100 \times \frac{27}{1000} \times 100\% \\ &= 270\% \end{aligned}$$ Equation 14

Adding another 'any' rule to pay 10 credits on "any 1 A", we have combinations {A,#A,#A}, {#A,A,#A}, and {#A,#A,A}. There are 243 hits due to this pay rule.

$$\begin{aligned} hits(any\ 2\ A) &= hits(A, \#A, \#A) + hits(\#A, A, \#A) + hits(\#A, \#A, A) \\ &= 1 \times (10-1) \times (10-1) \ + \ (10-1) \times 1 \times (10-1) \ + \ (10-1) \times (10-1) \times 1 \\ &= 81 + 81 + 81 \\ &= 243 \end{aligned}$$ Equation 15

The RTP for this rule is therefore 243%.

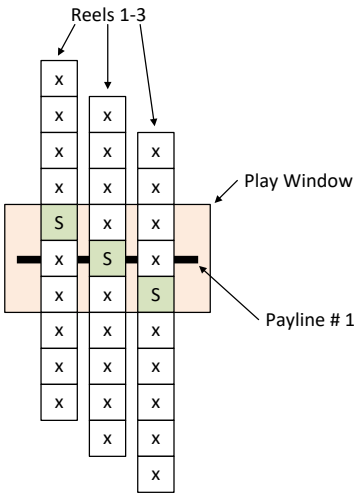
$$\begin{aligned} RTP(any\ 1\ A) &= prize \times \frac{hits(any\ 1\ A)}{cycle \cdot bet} \\ &= 10 \times \frac{243}{1000} \times 100\% \\ &= 243\% \end{aligned}$$ Equation 16

The RTP for the game is the sum of the RTP's of each of its pay rules, which for the 3 "any" rules is 613%

100%+270%+243% = 613%.

SCATTER

Normal, or non-scattered symbols, only pay (hit) when they are on the payline. A scattered symbol hits when it is visible in the play window anywhere on the reel, i.e. on, above or below the payline.



Consider what happens when the reel moves through the window, as the reel brings the scattered symbol (S) into each of the 3 possible positions (top, middle, bottom).

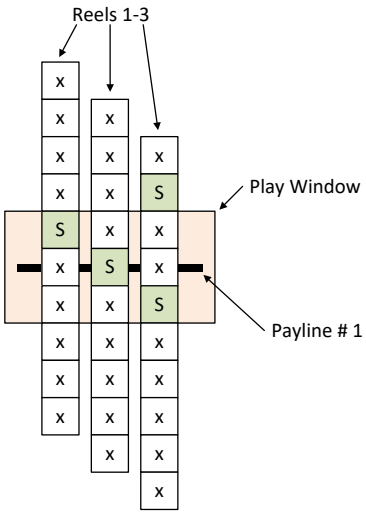
Rather than a total of one hit as the reel stops on the payline in each of its 10 possible positions we see that there are 3. The number of hits for the combination {S,S,S} is then

$$hits(S,S,S) = 3 \times 3 \times 3 = 27$$

And the hits for left 2 S, is

$$hits(S,S,\#S) = 3 \times 3 \times (10-3) = 63$$

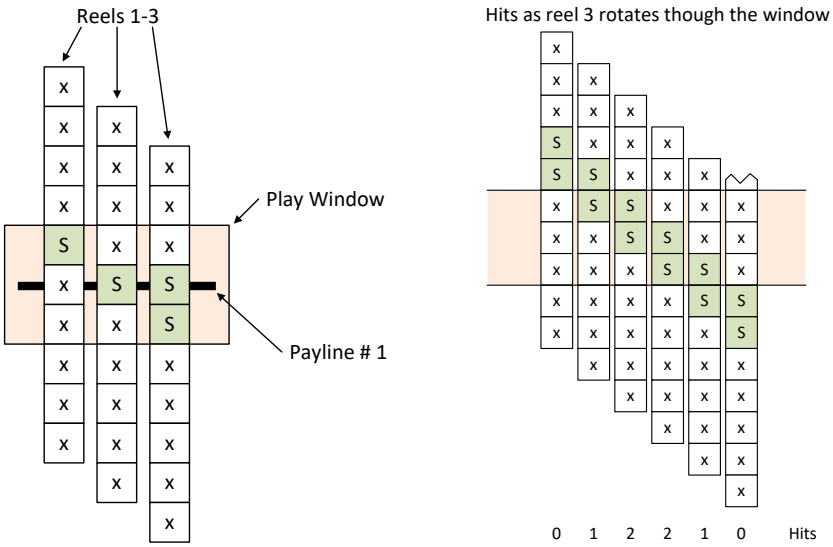
Note that if the window were 4 symbols high we would instead have 4 hits per reel, and 64 hits for left 3 S.



When we add a second scattered symbol to the reel we can see that there are 6 hits on reel 3 (2 × 2), hence the total number of hits for 3 S is 3×3×6=54.

Similarly for the combination {S,S,#S} there are only 4 stop positions on the last reel where the S does not hit (i.e. is not visible in the window), and we have 3×3×4 = 36 hits.

In the next example we again have 2 × S symbols on reel 3, but they are stacked on top of each other. Looking carefully at reel 3 as the reel rotates through the play window we can see that there are only 4 stop positions that causes hits instead of the 6 we had before with the scattered symbols spread out. However 2 of these 4 stop positions have 2 hits each instead of 1, so the total number of hits over the reel remains the same (6).



As the total hits over the cycle remain the same the RTP is unchanged. However the total wins awarded by the games are different. When a win is due to the 2 stacked symbols the number of hits doubles, hence the total win for that prize is also doubled. While the RTP is the same, the volatility is slightly changed.

Previously we counted 4 hits of #S with 2 separated scatter symbols on the reel, but now there are 6 hits with no compensating change in the number of hits at each – these remain zero. Hence we can see that the reelstrip layout can change the RTP of games that have scatter symbols. Normally scatter symbols are separated on the reels, so that multiple scatter symbols cannot be seen on the same reel at once.

When designing games, especially using Excel, the calculations are typically performed assuming scatters are spread out on the reels so that this effect does not occur, and when the actual game is built the symbols are spread out to match this assumption. In the event that the scatters are not properly separated an error will result.

In Slot Designer a symbol can be globally scattered with the scatter command.

```
scatter A;
```

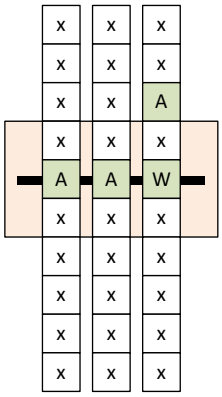
We can also scatter it only where it is used in a combination by adding ".s" after the symbol.

```
pay 100 on left 3 A.s;
```

WILD SYMBOLS

A wild or substitute symbol is one that can be used in place of other symbols in the pay table combinations.

Consider the game of Figure 1 (page 4), where one of the X symbols is replaced by the wild symbol 'W' on reel 3.



For the combination {A,A,A} we have 1 hit on reels 1 and 2, but on reel 3 both the A and W will hit for this combination, hence we now have 2 hits where before we had only one.

$$hits(A,A,A) = 1 \times 1 \times 2 = 2$$

An alternate notation to represent this combination, embedding the wild, is

{A,A,{A,W}}

This indicates that A or W is counted on the 3rd reel of the payline.

This is equivalent to adding a second A to reel 3, however if we had other symbols and combinations, the wild symbol could substitute for other symbols than just A.

One well known variation on substitution is to double the prize when a wild is used to match the combination, so that {A,A,W} would pay double the prize of {A,A,A}. This simply requires counting the hits for the various combinations, and determining the RTP from the appropriate prize values.

The RTP of this game where 3 × A pays 100 credits, W is wild and doubles the prize, and the bet is 1 credit, is 30%.

$$\begin{aligned} hits(A,A,A) &= 1 \times 1 \times 1 = 1 \\ hits(A,A,W) &= 1 \times 1 \times 1 = 1 \\ RTP(A,A,A) &= \frac{hits \times prize}{cycle \times bet} \times 100\% = \frac{1 \times 100}{1000 \times 1} = 10\% \\ RTP(A,A,W) &= \frac{1 \times 200}{1000 \times 1} = 20\% \\ Total\ RTP &= 10\% + 20\% \\ &= 30\% \end{aligned}$$

Equation 17

In Slot Designer substitution can be implemented in several ways. We can include the wild symbol 'W' in the pay rules

```
pay 100 on left 3 { A, W };
```

Alternately we can use the substitute command to automatically modify all the combinations to include the wild symbol.

```
substitute W = all;
pay 100 on left 3 A;
```

The wild symbol does not usually apply to the scattered symbol 'S'.

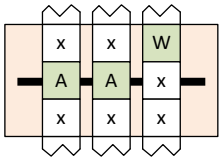
```
substitute W = all except S;
```

We can automatically double the prize value when the wild symbol is used.

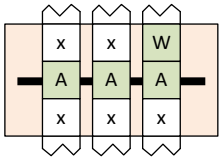
```
substitute W = all except S pay = pay * 2;
```

Expanding Wild

A popular variation on substitution is to scatter the wild symbol, so that if the wild is anywhere on the reel it substitutes into the payline combination. Normally the wild symbol substitutes only when it is on the payline. In this example the combination {A,A,A} hits due to the off-payline scattered wild symbol W.



As with other types of scattered symbol this can result in multiple hits on the reel when the combination symbol is on the payline (1 hit) and the wild symbol is above or below the payline (1 more hit).



Given that an expanding wild usually displays an animation that expands to cover the entire reel and payline combination symbol cannot be seen, the hit from the payline symbol is usually not counted – i.e. we count 1 hit, not 2. We can then use the RTP contribution saved in another part of the game where it will be visible to the player.

In Slot Designer expanding wild, where the wild symbol W substitutes for A and W is scattered, and where we count multiple (2) hits for both A on the payline and scattered wild off the payline, can be implemented as

```
pay 100 on left 3 { A, W.s };
```

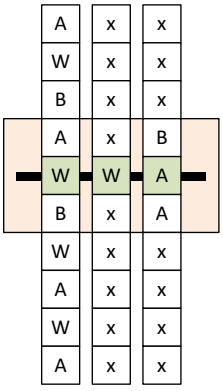
When we want to reduce the multiple (2) hits to a single hit we use the merge attribute

```
pay 100 on left 3 { A, W.s }.merge;
```

PRIORITISATION

Consider a game where the wild symbol W substitutes for symbols A and B, and the reels stop with symbols {W, W, A} on the payline. This matches both left 3 A and left 2 B pay rules (combinations {A,A,A} and {B,B,#B} respectively).

While we could pay both, it is more common to pay on the highest win only, hence the common rule on slot games of "Highest win pays on each payline". This is also referred to as prioritization, as we order the priority of the pay rules, and determine which single rule to pay.



Left 3 A pays 200

Left 2 B pays 100

The hits without prioritization, including wild, are

$$hits(A, A, A) = (4 + 4) \times 1 \times 2 = 16$$ Equation 18

$$hits(B,B,\#B) = (2+4)\times 1\times(10-1) = 54$$ Equation 19

The RTP for the non-prioritized game is then 860%

$$rtp = \frac{16\times 200 + 54\times 100}{1000\times 1}\times 100\% = 860\%$$ Equation 20

Consider the winning combination {W,W,A}. When all wins pay this would hit both left 2 B and left 3 A. When only the highest win pays then we need to decide to pay either the 2 B or 3 A, and choose 3 A as it's the highest (otherwise the player would be quite upset). We need adjust, or discount, the hits for left 2 B to exclude those hits that pay as left 3 A instead.

The combinations where we pay left 3 A instead of left 2 B are those where stop with {W,W,A} on the payline, for which there are 8 hits

$$hits(W,W,A) = 4\times 1\times 2 = 8$$ Equation 21

The number of hits on left 2 B is then 54 - 8 = 46 hits.

The RTP for the game is 780%, a considerable drop from the non-prioritized 860%.

$$rtp = \frac{16\times 200 + 46\times 100}{1000\times 1}\times 100\% = 780\%$$ Equation 22

Where pay rules have the same prize value the priority is chosen by the game designer.

In a real game using Excel the discounting gets quite involved.

Prioritization in Slot Designer is implemented as

```
prioritise highest
{
    pay 200 on left 3 A;
    pay 100 on left 2 B;
}
```

MULTILINE

Most games have more than one payline, and some have 100 or more. There is no standard layout for paylines, apart perhaps from the first 3.

The number of paylines and their layout is usually of little concern mathematically, as the RTP tends to be independent of the number of paylines in the game; in Excel the RTP is usually calculated for a single payline only. The hits and therefore win increases linearly with additional paylines, hence the bet is similarly increased to keep the RTP constant.

As hits for scatter combinations are independent of paylines their prizes are multiplied by the bet to keep the RTP constant.

The maximum number of paylines in a game is the product of the individual window heights for each reel., and in games where it is used is sometimes described as "ways to win". In a 5 reel 3 row games this is 3^5, or 243 lines or ways to win. Similarly for a 5 by 4 game there's a maximum of 4^5 or 1024

paylines. A game with a window of 3, 4, 5, 4, 3 symbols visible on reels 1 to 5 respectively would have a maximum of 3×4×5×4×3=720 paylines.

The 5 paylines shown above were created by the following Slot Designer code

```
payline line01 { 2,2,2,2,2 }                    // define the lines
payline line02 { 1,1,1,1,1 }
payline line03 { 3,3,3,3,3 }
payline line04 { 1,2,3,2,1 }
payline line05 { 3,2,1,2,3 }
payline line01, line02, line03, line04, line05;     // instantiate the paylines into the game
```

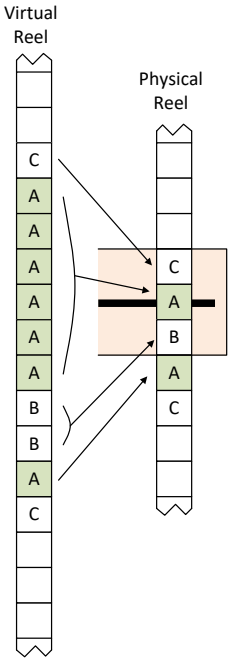
VIRTUAL REELS

Virtual reels were originally used in mechanical stepper games to implement symbol weightings, as described in the Telnaes patent (US 4448419). While not common, some game designers also like to use them in video slots.

In a mechanical stepper machine the reel is a physical device, and mechanical constraints on the size of the reel and symbols limit the number of symbols around the reel to about 22. If there were 100 symbols per reel for example, then each symbol would be so small the player wouldn't be able to see them, or conversely the gaming machine would be enormous to fit the massive reels.

Considering a 3 reel game this limits the cycle to 22^3, or 10648 different positions. If we wanted a 20K credit prize, and only one combination of the 10648 games hit, we would still have an RTP from that one prize of around 188%. This severely limits the maximum prizes on the pay table, and flexibility in designing the game.

Virtual reels model a longer virtual reel, which maps multiple virtual stopping positions to a single physical stop position on the actual reel.



In this example we have a virtual reel with one stack of 6 × A symbols and another 1 × A symbol. The virtual symbols map to the physical reel, so that when the virtual reel is spun and a stop position is determined, it determines a corresponding stop position on the physical reel.

When we spin the virtual reel we can see that the chance of a hit on the stack of 6 A symbols is 6 size times higher than hitting the single A symbol, and also 3 times that of hitting the B symbol. Hence we can see that the weighting of the symbols on the virtual reel affects the probability of stopping on a symbol on physical reel.

Consider again the previous example of a 3 reel game, with a single combination that wins 20K credits. We could easily make a virtual reel of 100 symbols, giving a cycle of 100^3 = 1 million games. The RTP due to that prize is now only 0.2% instead of the 188% we had without the virtual reel.

On stepper machines virtual reels allow vastly increased prizes, and flexibility in pay table design.

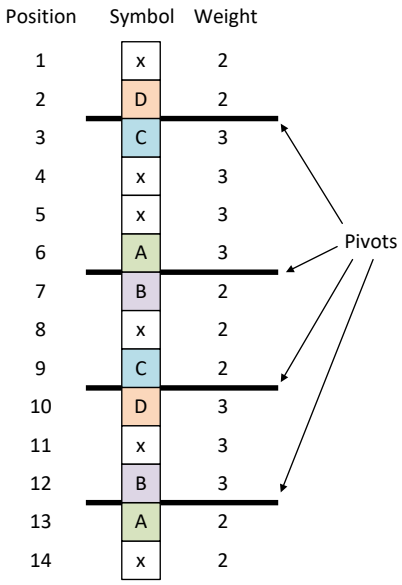
Unlike normal games, the win (hence RTP) of each combination is different for each payline, greatly complicating Excel spread sheets. To understand why this happens consider a virtual reel with a very large weighting at one position. When the reel spin stops, it is highly likely that this symbol will stop on payline 1, and correspondingly highly unlikely that it will ever stop on the position above or below that (i.e. on another payline).

It is possible to balance the symbols on the reel so that the RTP for each payline is the same.

The symbols must be carefully arranged on the reel such that there are pivot points where the symbols directly above and below each pivot point are repeated around another pivot point, and the weights balance each other.

In the diagram to the right the pair of symbols A, B at positions 6, 7 around the pivot have weights of 3, 2 and are balanced by the pair B, A at positions 13, 14 with weights 3, 2.

Only the symbols next to the pivot points require balancing.

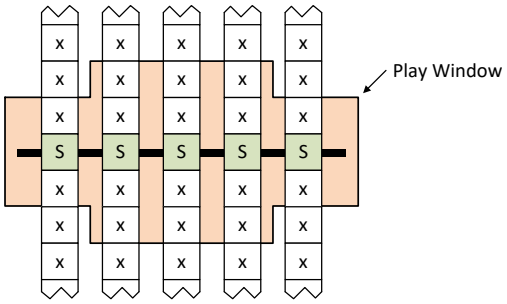


It may not be possible to balance the symbol layout, for example when converting a video game to a stepper format. In this case it may be considerable amount of work to not only calculate the RTP for each payline, but where the player may select different numbers of paylines to play, to keep the RTP for each player selection approximately the same. It may be a regulatory requirement to have the RTP the same or increase as the number of paylines increase.

IRREGULAR WINDOW

Most games have the usual 3x3 or 5x3 window, but some games have more unusual shapes. The RTP for these games is easily calculated using the same principles outlined so far. The normal payline combinations are independent of the height of the window; it is only scatter wins that are sensitive to window height.

This example shows a window with 3,5,5,5,3 symbols visible on reels 1 to 5 respectively.



For example the number of hits for a scatter symbol on all 5 reels is 1125.

$$hits(S,S,S,S,S) = 3\times5\times5\times5\times3$$
$$= 1125$$

In Slot Designer the window is described with the following code, which first defines the maximum size of the window, then defines which locations within it are used.

```
window 5,5,{2,3,4},{1,2,3,4,5},{1,2,3,4,5},{1,2,3,4,5},{2,3,4};
```

PLAYER CHOICE

A feature of some games is to present the player with a set of choices (e.g. boxes) with hidden prizes and ask them to pick one or more of them. As each prize is chosen the value of the prize is revealed, and the player wins the sum of the prizes chosen. These are also known as pick games.

For this simple case the average win is

$$average\,win = number\,of\ \ choices \times average\,prize\,value$$ Equation 23

There are many variations of these types of features, such as picking other feature games, multipliers, or prizes that immediately terminate the choices. Some are quite complex, and are beyond the current scope of this document.

FREE GAMES

A common game feature is free spins, where a base game triggers into a series of free games.

Consider first the case where free games do not retrigger additional free games. The average win of the base game and free game is B_0 and B_1 respectively, and the base game triggers n_1 free games with a trigger probability of p_1. Hence the average number of free games is n_1p_1

$$E(\,free\ games) = n_1 p_1$$ Equation 24

The final win, F, of the combined series of base and free games is then

$$F = B_0 + n_1 p_1 B_1$$ Equation 25

As usual the RTP is the win divided by the bet.

$$final\ rtp = \frac{B_0 + n_1 p_1 B_1}{bet} \times 100\%$$ Equation 26

Now consider what happens when the free game is able to retrigger n_2 games with probability p_2. The average number of free spins is

$$\begin{aligned} E(\,freespins) &= n_1 p_1 + n_1 p_1 (n_2 p_2) + n_1 p_1 (n_2 p_2)^2 + n_1 p_1 (n_2 p_2)^3 + \ldots \\ &= n_1 p_1 \sum_{i=0}^{\infty} (n_2 p_2)^i \\ &= n_1 p_1 \left(\frac{1}{1 - n_2 p_2} \right) \end{aligned}$$ Equation 27

And the RTP is again the base win plus the average free game win times the average number of free games, divided by the bet.

$$final\_rtp = \frac{B_0 + B_1 n_1 p_1 \left(\frac{1}{1 - n_2 p_2} \right)}{bet} \times 100\%$$ Equation 28

STANDARD DEVIATION

The standard deviation σ is calculated (Equation 29) using the probabilities of each prize in the base and feature games, times the difference between that prizes credit value and the mean credit win of the entire series of base/feature games.

$$\sigma = \sqrt{\sum_{i=1}^{n} p_i (x_i - \bar{x})^2}$$ Equation 29

The standard deviation suitable for regulatory approval can be calculated in Excel. The probability of each prize in a feature game is the total probability of reaching that feature game times the probability of that prize being awarded within the feature game.

The probability of the base game itself is 1.0, and the probability of the free game is the expected value of the free spins as calculated by Equation 27.

We should include the hits with zero wins (i.e. misses), but this cannot be calculated analytically in Excel for games with coinciding wins (most, if not all of them). We can leave them out or make an approximation that it's the cycle minus the sum of the hits, but either way it's wrong. In a related issue, we normally only model a single payline in Excel the model, and hence the standard deviation is for one line even when we have a multi-line game, and the multiline game will have even more coinciding wins than a single line game.

Further, using Excel we can only calculate the non-coinciding wins, but the player experiences coinciding wins. i.e. when they player wins 100 and 200 credits due to simultaneous scatter and line wins, do we consider this as a single win of 300 credits or separate wins of 100 and 200 credit? The player experiences it as 300 credits, but we can't calculate it theoretically. We could perform a full cycle simulation to determine the correct distribution of wins, or a random simulation to obtain an approximation.

When we win free games, we have coinciding wins across the series of game played, for which we could also calculate a standard deviation. We can only perform a random simulation to estimate this type of standard deviation.

There are several ways to calculate the standard deviation, and which you use depends on what you're trying to achieve.

VOLATILITY INDEX AND CONFIDENCE INTERVALS

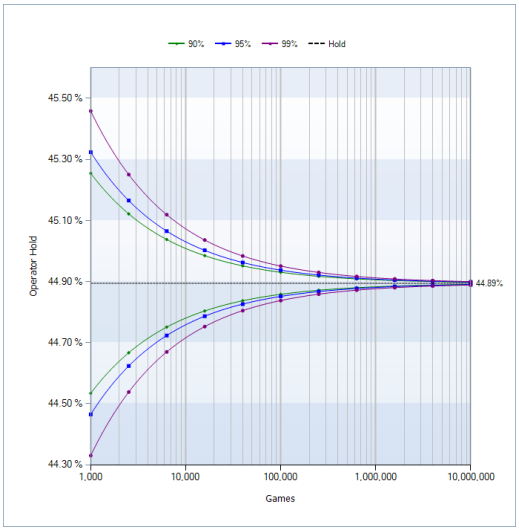
The confidence intervals show the range of the RTP on a gaming machine after N games

$$rtp\ range = rtp \pm \frac{volatility\ \ index}{\sqrt{N}}$$ Equation 30

Where the volatility index based on a 95% confidence interval is

$$volatility\,index = 1.96 \cdot \sigma$$ Equation 31

If the measured RTP is outside this range then, with 95% confidence, we can say it is not working as expected. This calculation assumes the game has a normal distribution, which may not be the case.



Slot Designer displays the confidence levels to show the expected margin to the game operator, to 90, 95, and 99% confidence levels.

The volatility index based on 90 and 99% confidence intervals is 1.64 σ and 2.58σ respectively.

CHAPTER 3 243 WAYS GAMES

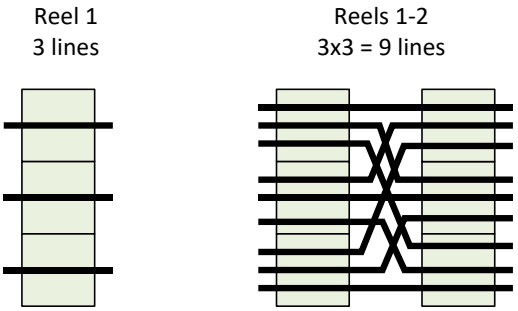
[1]Slot games have historically used paylines to determine winning combinations of symbols across the play window. Originally a single payline passed through the center of the window, but over time the number of paylines has increased to 100 and more. As the number of paylines has increased it has become progressively more difficult for players to understand why they won a particular prize.

By removing the concept of paylines entirely, 243 ways games make it relatively easy for the player to understand why a particular combination wins. They simply count the number of symbols appearing on the reel, typically from left to right, without regard to their position on the reel. For example, if 3 ACE symbols appear anywhere, one on each of the first 3 reels, they win the 3 ACE combination without having to consider if it's on a payline or not.

WAYS VS. PAYLINES

A 243 ways game is so called because there are up to 243 "ways" to win, but it's not a 243 payline game.

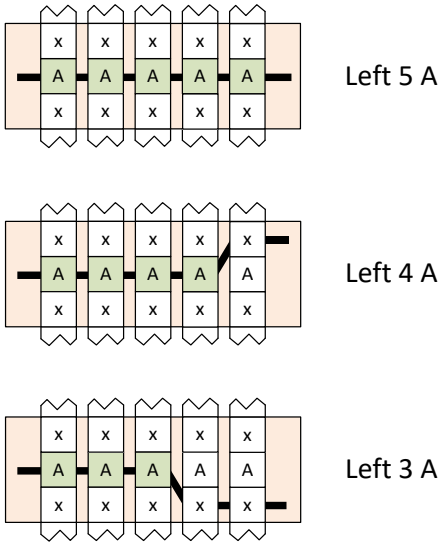
There are a maximum of 243 possible paylines in a 5×3 game, as can be understood from the following diagram showing all the paylines for the first 2 reels. For a single reel there are 3 possible paylines, and for every additional reel each payline splits into 3 further paylines, hence 3, 9 (3×3), 27 (3×3×3), 81 (3×3×3×3), and 243 (3×3×3×3×3), for 1 to 5 reels respectively.



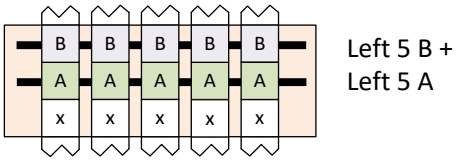
We can see that 243 ways is not 243 paylines by considering the following diagram. According to the basic idea behind the 243 ways game, 5 A symbols anywhere on the 5 reels counts as a single 5 of a kind win. However, if we evaluate this as a 243 line game we can see that it is counted as at least 3 coinciding wins of 3, 4, and 5 A. Therefore, we can think of 'ways' as being similar to lines, but where a win occurs on a line a subset of those symbols cannot be used by themselves to make a win on another line.

[1] This chapter was contributed by Mark Sinosich of Imagine Numbers

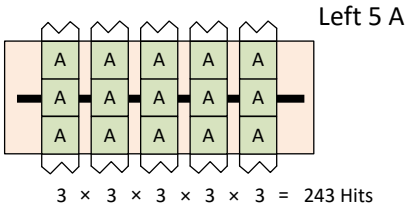
Figure 2.



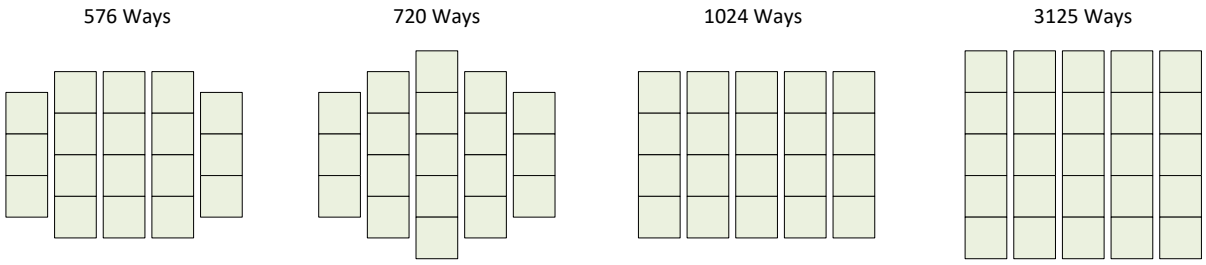
We cannot prioritise the wins to select only a single win only for the highest number of symbols, as can be seen from the next diagram where there are valid wins on multiple paylines.



We can however treat the 243 ways game as a 1 (or 0) line game where all the symbols are scattered, and this provides a simple means of calculating the RTP provided that there are no wild symbols on the leftmost reels.



By varying the window size we can see different numbers of ways are possible (e.g. 3×4×4×4×3 = 576).



BET

Payline games usually have a bet of 1 credit per payline, but 243 ways games typically have a bet in the region of 25 to 30 credits. This value is a balance between having a large enough bet to get reasonable size wins, and not having such a large bet that the game is too expensive to play.

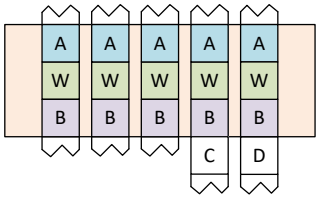
It is generally considered undesirable to have wins smaller than the bet, as these wins are in fact losses. While a bet around 30 credits is quite high, 243 ways games tend not to have 2 of a kind wins with their associated low prize values (due to the large number of hits), and do have a significant number of coinciding wins, thus reducing the potential for wins less than bet.

WILD ON REEL 1

When we have wild symbols on the leftmost reels the determination of the win is considerably more complex. In the following two examples the symbol W is wild for symbols A, B, C, D, and A pays more than B, B more than C, and C more than D.

In the next diagram there are 32 (2^5) possible paylines through the top 2 rows of the window, and given that W is wild for A, we have 32 winning paylines for 5A. There are also 32 paylines through the bottom 2 rows, and W is wild for B, so we have 32 winning paylines for 5B. However the A and B wins share a single payline through the center row, and as this is a 'ways' game only one of the combinations will pay, in this case A since its prize is higher than that of B. Hence there are only 32 – 1 = 31 wins for the 5B combination.

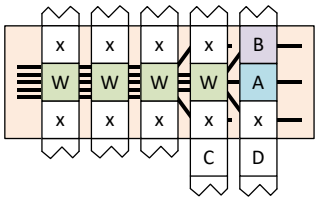
Given that W is wild for symbols C and D, the center line also matches the 5C and 5D combinations, but only the higher paying 5A is awarded.



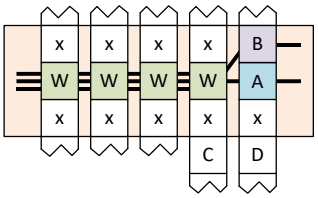
32 × left 5 A +
31 × left 5 B

The overlap in the combinations for the different symbols is caused by the wilds on the leftmost reels, and it is this overlap that prevents the calculation of the win by scattering symbols, as we would count 32 hits on each of A, B, C and D. Removing the wild from the leftmost reels (usually reel 1) would eliminate the overlap, simplify the calculations, and significantly reduce the number coinciding hits.

In the next example with 4 wild symbols there are multiple ways we can interpret the win. On the left diagram, wins of 5A, 5B, 4A, and 3A (the 4W symbols match all of 4A, 4B, 4C and 4D, but only the highest, 4A, is awarded). Note however that this contradicts the example above showing how 243 ways games differ from line games, for this special case of subsets of wilds causing wins. On the right we can interpret the rules to mean there are no wins due to the subsets of the winning wild symbols. From a game design view point we may prefer the rules on the left as it is more generous to the player. However it would be even better to avoid this situation completely as it is confusing to the player.

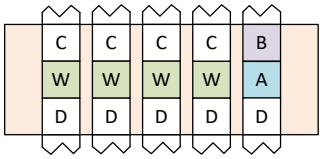


1 × left 5 A +
1 × left 5 B +
1 × left 4 A
2 × left 3 A



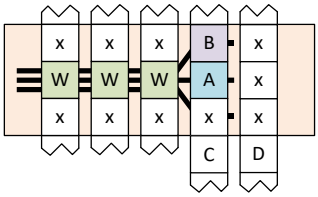
1 × left 5 A +
1 × left 5 B

In the above example we have the non-paying symbol 'x' for which wild does not substitute, for example a scatter or bonus, and would typically not be placed next to a wild symbol. Using normal paying symbols, and considering only 4 and 5 of a kind wins, for both of the above scenarios we would then have the following.



1 × left 5 A +
1 × left 5 B +
16 × left 5 D +
15 × left 4 C

By moving the symbols on reel 5 to reel 4 (and using the rule interpretation on the left above) the wins in the next example are similar, but note how the 3 winning paylines at reel 4 would each split into a further 3 (i.e. 9 in total) on reel 5, but there are no further wins due to the subset rule (Figure 2, page 18).



1 × left 4 B +
1 × left 4 A +
1 × left 3 A

We can calculate the RTP of these games (using the rule interpretation on the left above) by calculating the hits as usual, then removing any hits on the wilds that would be awarded to another symbol. For example the hits for 5A are calculated for the combination

hits(5A)= hits({{A.s,W.s}, {A.s,W.s}, {A.s,W.s}, {A.s,W.s}, {A.s,W.s}})

The hits for 5B are similar, but we must account for the hits due to 5W being attributed to 5A instead of 5B

hits(5B)= hits({{B.s,W.s}, {B.s,W.s}, {B.s,W.s}, {B.s,W.s}, {B.s,W.s}}) – hits({W,W,W,W,W})

Similarly hits for 4A are

hits(4A)= hits({{A.s,W.s}, {A.s,W.s}, {A.s,W.s}, {A.s,W.s}, #{A.s,W.s}})

And the hits for 4B are similar, minus the hits for 4W which are attributed to 4A instead of 4B

hits(4B)= hits({{B.s,W.s}, {B.s,W.s}, {B.s,W.s}, {B.s,W.s}, #{B.s,W.s}}) – hits({W,W,W,W,#W})

RANDOM WINDOW SIZE

A popular class of games vary the window size randomly per spin.

Consider the simple case where the window height for each reel varies randomly and independently between 2 and 3 symbols. Each reel has two possible sizes, and there are 5 of them, so we have 2^5 = 32 possible window sizes. To a first approximation we would have at least 32 different games to consider, and 32 sheets in a spreadsheet.

In more realistic example there might be 6 reels, each of which randomly varies between 2-7 symbols high. Hence each reel has 6 different sizes and the total number of window size we need to consider is 6^6 = 46,656. The largest of these windows is a game with 7^6 = 117,649 ways. There are, hopefully, simplifications that can be made to reduce the size of the model.

These games result in very large spreadsheets that can take minutes to open in Excel. In Slot Designer a simple example with 46,656 games is about 60 lines of code and around 10 seconds to analyse. i.e. creating the game in Slot Designer is quicker than opening an existing spreadsheet.

CHAPTER 4 METAMORPHIC GAMES

There doesn't appear to be any precise definition of what metamorphic means, but generally it's a game that evolves as play progresses. i.e. the current game state is not independent of previous games.

Following this (imprecise) definition, a progressive could be considered as a metamorphic game, the progressive jackpot increases during play and the game changes rather dramatically if you win the jackpot, but it's not. It's a progressive.

A better example would be the popular "Sticky Wild", where during the free spins a wild symbol landing anywhere in the window will "stick" to that position and take part in wins for the remainder of the free games. The game evolves as it plays.

A 5×3 Sticky Wild game has 15 possible symbol locations on the window, and if each one can have a stuck wild or not that's 2^{15} = 32768 variations of wilds on the window, each one of which has a different win. At a first approximation a spreadsheet would therefore require at least 32768 sheets to model this game (and note that a 5×4 game has 1,048,576 variations). This is clearly impractical, and the mathematician needs to make simplifications to model these games. Errors will occur when the simplifying assumptions are incorrect.

These games tend to be highly complex and require a high level of mathematical ability to design with Excel. I've not attempted to explain it in Excel, I don't have the time.

PERSISTENT STATE GAMES

Persistent state games have some element of the game that persists across plays of the base game. These games are inherently metamorphic as they evolve during play.

As usual there doesn't appear to be any precise definition. Progessives could be (but are not) considered as having persistent state as the accumulation of the jackpot continues across plays – it's not reset each time the player presses the play button.

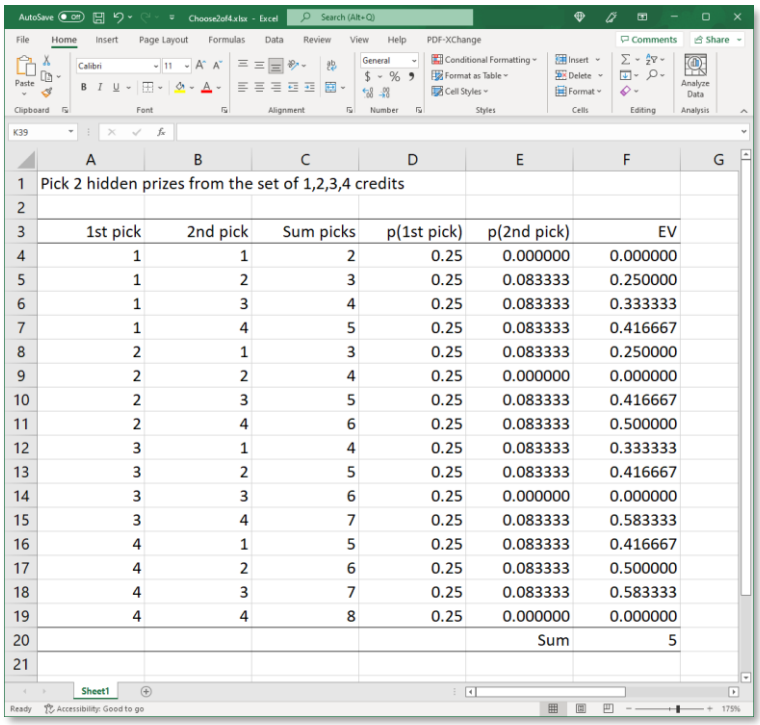
A common example would be where an event in the game increments (accumulates) a counter and when a limit is reached a feature is triggered.

CHAPTER 5 FEATURES WITHOUT REELS

Many features are variants of picking or choosing prizes from a set of boxes on the screen, with the prizes being randomly distributed and hidden; you don't find out what you've won until after you've chosen them. There are *many* variations on this theme.

Using an example of 4 boxes with prizes of 1,2,3,4 credits, what is the average win when we pick two of them?

Picking the first box we have a probability of 1 in 4, or 0.25 of picking 1,2,3, or 4. After that there's zero probability of picking the same prize again and only 3 prizes remaining, so we've a probability of 1 in 3, or 1/3 of picking any of those.



	A	B	C	D	E	F	G
1	Pick 2 hidden prizes from the set of 1,2,3,4 credits						
2							
3	1st pick	2nd pick	Sum picks	p(1st pick)	p(2nd pick)	EV	
4	1	1	2	0.25	0.000000	0.000000	
5	1	2	3	0.25	0.083333	0.250000	
6	1	3	4	0.25	0.083333	0.333333	
7	1	4	5	0.25	0.083333	0.416667	
8	2	1	3	0.25	0.083333	0.250000	
9	2	2	4	0.25	0.000000	0.000000	
10	2	3	5	0.25	0.083333	0.416667	
11	2	4	6	0.25	0.083333	0.500000	
12	3	1	4	0.25	0.083333	0.333333	
13	3	2	5	0.25	0.083333	0.416667	
14	3	3	6	0.25	0.000000	0.000000	
15	3	4	7	0.25	0.083333	0.583333	
16	4	1	5	0.25	0.083333	0.416667	
17	4	2	6	0.25	0.083333	0.500000	
18	4	3	7	0.25	0.083333	0.583333	
19	4	4	8	0.25	0.000000	0.000000	
20					Sum	5	
21							

We can build a simple spreadsheet to model this game by enumerating all possible choices.

We have the 1st and 2nd pick prizes (columns A,B), the total win (column C), the probability of the 1st and 2nd picks (columns D,E), and the average win of each set of picks (column F).

The probability of the 2nd pick is zero where its not possible, and ¼*1/3 = 0.083333 where it is. The average win of this feature is then 5 (cell F20).

CHAPTER 6 PROGRESSIVES

This section is described mainly in terms of land-based casino progressives, but the principles are the same for other forms.

The classic progressive jackpot is a large jackpot prize contributed to by game play across a number of gaming machines, and being won infrequently and at random by one of the players.

Casino based progressives range from standalone progressives within a single gaming machine, progressives across a bank of machines within a casino, multi-property and state-wide (IGT's Megabucks). The more connected games the faster the jackpot increases and the higher the jackpot prizes, with IGT's Megabucks reaching into the tens of millions of dollars.

TURNOVER AND INCREMENT

As a game is played it sends its turnover data to the progressive controller, which apportions some of this to the jackpot pool. This portion, called the increment, is split among several jackpot pools in a multi-level progressive.

The turnover is the amount bet on the game. E.g. if the game cost $1.00 per spin, the turnover is $1.00 every time the game is played.

For an increment of 10% the jackpot controller increases the progressive pool by $1.00 * 10% = 10 cents for each game played. In a sense the jackpot controller is giving money away, but in practice the RTP of the game is reduced to compensate for the increase due to the jackpot controller. Typically games are designed with a range of RTP's to match the different jackpot controllers they will be used with.

Another way of thinking of the increment is that it is money that has not yet been won.

STARTUP

When a progressive is won (or first started) the pool prize value is reset to the start-up value. The start-up value is typically non-zero; otherwise the progressive would not be very appealing to play just after a payout.

The frequency of triggering the jackpot has no effect on the RTP due to the increment, changing only the maximum value the progressive pool reaches. This is because the increment is always money that will be awarded back to the players, the only question is when.

However the startup does affect the RTP, as can be seen by considering two extreme cases. First, consider where the start-up is $100 and the progressive triggers once in every 1 million games, each of which costs $1.00. The total bet is $100M to win the startup of only $100. The RTP is clearly very low. Second, consider the progressive triggering every second game, where for a cost of only $2.00 the player wins $100; the RTP is clearly very high.

PROGRESSIVE TRIGGERS

There are two basic types of progressive, Mystery and Game Triggered, defined by where the win determination is made.

In a Mystery progressive the win is determined by the progressive controller. When the progressive starts, and following a payout, a secret (hence mystery) random value between the start-up andmaximum progressive value is chosen, and when the progressive pool later crosses that threshold the pool is awarded to the player that caused it.

In a game triggered progressive the game determines the progressive win, sending a signal to the progressive controller indicating the win. In the event that two games signal a win at the same time

the first to reach the progressive controller wins the pool, the pool is reset to the start-up value, and the second wins the start-up value.

Another difference between the two types of triggers is that the mystery progressive is guaranteed to hit, i.e. it is deterministic, while the game triggered jackpot is not.

A multilevel progressive maintains multiple prize pools, with separate triggers and increments (of turnover).

MYSTERY PROGRESSIVE

The average trigger point of a mystery progressive is the midpoint between the startup and the maximum (as it's randomly selected between these two limits).

$$\text{average trigger} = \frac{\max - startup}{2} \qquad \text{Equation 32}$$

Hence the average turnover required to trigger a win is

$$average\ turnover to\ win = \frac{\text{average trigger}}{increment} = \frac{\left(\frac{\max - startup}{2}\right)}{increment} \qquad \text{Equation 33}$$

The average win is

$$average\_win = \frac{\max + startup}{2} \qquad \text{Equation 34}$$

The RTP is the average win divided by the average cost (turnover to get the win)

$$rtp = \frac{\frac{\max + startup}{2}}{\left(\frac{\max - startup}{2}\right) \Big/ increment} = \frac{\max + startup}{(\max - startup) \cdot increment} \qquad \text{Equation 35}$$

GAME TRIGGERED PROGRESSIVE

The game triggered progressive RTP is the sum of the startup RTP and increment.

$$\text{progressive rtp} = \text{startup probability} \times \text{startup value} + increment \qquad \text{Equation 36}$$

As the jackpot is entirely triggered by the game, the startup probability is defined by some event in the game. We might for instance trigger the progressive by having 5 jackpot symbols on the payline, and the probability is determined as discussed previously.

CHAPTER 7 SIMULATION VS THEORETICAL MODELLING

The two common approaches to mathematical game design, theoretical probability-based modelling (typically Microsoft Excel) and simulation of an game implementation (Java, Python, C++, C#, etc). There are pros and cons to each.

- Excel cannot model all games, simulation can.
- Simulation acts like a black box and it can be difficult to understand where game statistics come from. Its relatively easy with Excel.
- Excel requires a higher level of mathematical skill.
- Excel cannot calculate coinciding wins and some game statistics are not available. Simulation can calculate these statistics.

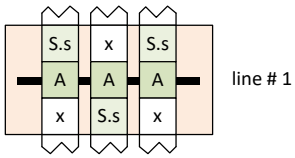
Many companies use both Excel and simulation together to design their games so that they can be checked against each other before the product is implemented. The commercial cost of bugs increases significantly the later in the design process we find them, and the added upfront cost of developing two models is outweighed by the savings later on. It's also an advantage to have two models built in a very different ways as they are less likely they share common design errors producing the same wrong result.

Slot Designer incorporates elements of both Excel style modelling and simulation, plus a considerable number of unique features. The original Slot Designer was released in 2008 with the language now termed SD3. The next generation, Slot Designer 2022, supports both SD3 and the new language and mathematics of SD4.

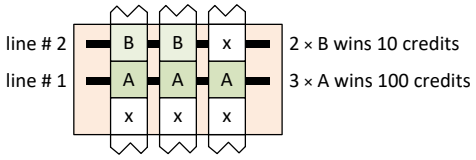
7.1 COINCIDING WINS

The calculation of RTP in Excel is a theoretical calculation based on the probability of combinations, and produces statistics for each combination or win independently. We cannot determine the probability of wins occurring on multiple paylines at the same time, or a payline win combined with an off-payline scatter win.

For example, we can calculate the probability of a win due to 3 × A, and 3 × scattered S independently, but not the probability of getting both at the same time.



Similarly we can calculate the probability of wining 100 credits and 10 credits independently, but not the probability of winning 110 credits.



A win that comprises multiple independent wins is referred to as a coinciding win, and provides important statistics related to game performance. It gives a more accurate picture of the game from the player's perspective, and in some cases is required for regulatory compliance.

When a player wins on multiple paylines their experience is of winning the coinciding win, not the individual parts – i.e. they think they won 110 credits, not 100 and 10.

Coinciding statistics can be determined by creating a simulation of the game with a custom program, for example in C# or C++, so that for each reel stopping position in the game the coinciding win is determined and recorded. This not only provides the coinciding statistics, but can be used for checking the calculations in the spreadsheet before it is implemented on the gaming platform.

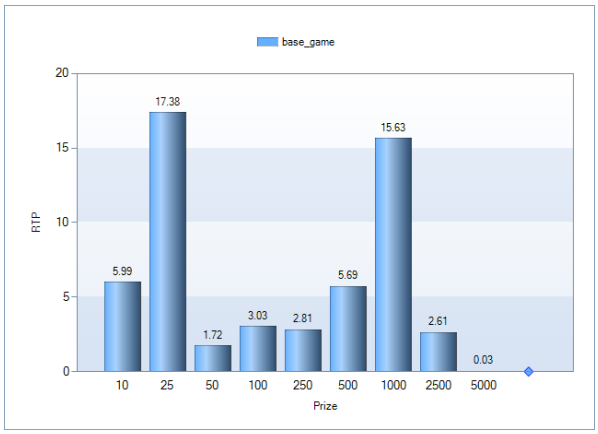
The simulation is typically done in two ways, either as an exhaustive cycle through all possible stop positions (referred to as full cycle), or using a random sampling of the possible outcomes (Monte Carlo). One or both methods may be used depending on the game and the required statistics. The full cycle simulation allows an exact match to the calculations in the spreadsheet, but is not capable of calculating coinciding statistics across a series of feature games. Monte Carlo simulation calculates coinciding wins across series of base/feature games, but does not provide a perfectly accurate result.

Slot Designer supports all these types of calculations and simulations.

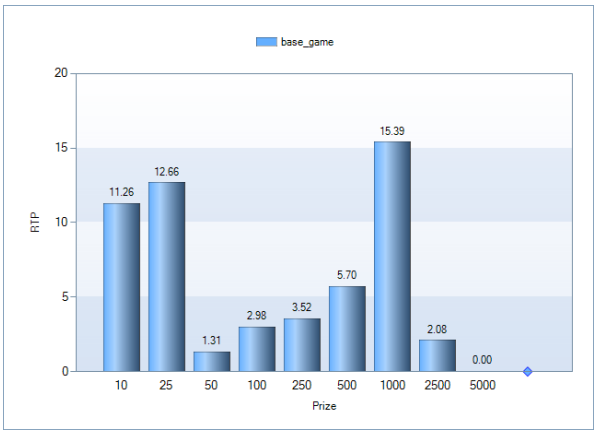
PROFILE

When we look at the profile of the game, dividing all the possible wins into ranges, we get a different picture of the game depending on the use of coinciding or non-coinciding statistics, and while it's easy in Excel to calculate the non-coinciding profile and produce such a chart, it is the coinciding data that is actually more useful.

The following charts show an example of difference between the results depending on how they are determined (these are not from a real game).



Coinciding



Non-coinciding

STANDARD DEVIATION

The standard deviation of the game reduces as we increase the number of paylines (it's less volatile), but using Excel we have no way of properly calculating the coinciding standard deviation. Fortunately it is not required for regulatory compliance.

MAXIMUM WINS

In some markets the maximum allowable win of the game is specified by government regulation.

The maximum usually results from a number of separate coinciding payline and scatter wins. For example the win may be limited to $10000, and while there is no individual win exceeding that amount we also need to ensure there is no possible combination of multiline wins and scatters wins over $10000 either. We probably also want to have a top pay table prize as close as possible to the limit as it's more appealing to the player, yet the closer we get to this prize for a single win the more likely we are to exceed it due to coinciding wins. This can involve numerous iterations of the game as we try variations of the pay table and reelstrip layout to optimize the game.

7.2 COMPLEXITY

As games become have become more complex the amount of work that goes into modelling them in Excel, and the skill required to do so, has increased dramatically. It can take weeks to months to build complex spreadsheet models, and some games that are not amenable to this kind of modelling at all. The well-known Cascading Reels games for example, can only be simulated.

Unlike theoretical modelling, simulation can be used to implement and design any game; if it wasn't we'd not be able to build the final product. Simulations are quick to build, generate coinciding wins, and it's much easier to find staff who can program than those who can build complex spreadsheet models. Why then, don't we use simulation for all games?

Excel has several advantages

- Its straightforward to understand where the games statistics come from, which is important during the design process, and when there's discrepancies in the game statistics between the model and platform implementation or regulatory test lab. Simulations are essentially black boxes where it's hard to understand what's going on inside, and while you'll know something is wrong it's very difficult to pinpoint what, where, and why.
- A theoretical model can calculate the exact game RTP with essentially no error. Random simulation always has an error and we're never quite certain its correct, so we increase the simulation size (and time) to reduce the error to within "acceptable" limits.
- Excel is quite interactive and its relatively easy to iteratively design the game by adjusting the parameters of the model and immediately view the resulting RTP. Doing the same with a simulation model tends to be slower.

Both approaches share the problem of validating complex models. It's hard enough to be sure even simple spreadsheets or programs are correct, let alone some of the vast models used in slot game design.

REDUCING COMPLEXITY

Ideally, we would reduce game complexity to speed up development and reduce costs, but that's wishful thinking; the market trend has been towards ever more complex games. Taking a closer look at what complexity actually means, there's two kinds of complexity[2] in game models.

- **Essential complexity** is that which is required to solve the problem. It is related to the problem itself and cannot be removed.
- **Accidental complexity** is not related to the problem, its wasteful, and it's something that we can, in principle, reduce or eliminate.

We can't reduce the essential complexity, but what about the accidental complexity? We commonly see this in software engineering where higher-level programming languages abstract away the details of the problem and let the user focus on the key issues. The less code we have the better.

Slot Designer aside, this hasn't worked work well for slot games, and the industry hasn't fundamentally changed since the advent of Excel[3] decades ago.

Both spreadsheets and simulation models shared similar issues, both of which are accidental complexity and fundamentally related to game tooling.

- They are general purpose tools not designed for modelling games and are inherently low level in the domain of slot mathematics. They have no concept of games, reelstrips, symbols, paylines, features, etc, and we must reproduce them every time we model a game.

---

[2] https://en.wikipedia.org/wiki/No_Silver_Bullet

[3] Excel launched in 1985, VisiCalc 6 years earlier, https://en.wikipedia.org/wiki/Spreadsheet

- It's very difficult, if not impractical, to package game functionality for reuse in other games. Each game is effectively a unique copy where any improvements we can make to the model won't affect any other games unless (and only when) they are copied from it. Libraries and code reuse are standard practice in software development, but game designers are typically not software engineers and are tasked with producing games, not game tools.

It's very clear the essential complexity of slot games has increased. While there are still many simple games being developed, the more complex modern slots are nothing like they were 20 or 30 years ago. What's not so clear is how much accidental complexity exists without something to compare it to.

Slot Designer directly addresses the issue of accidental complexity, aiming to raise the level of abstraction as far as possible.

Using a couple of specific examples, the first Slot Designer generation of (SD3) reduced the size of one popular game model by about 50 times, and the second generation (SD4) has reduced half a million lines of SD3 code to only 60 in SD4. In Excel these kinds of games take minutes just to open the file, and it's a lot easier to type in and find bugs in 60 lines of code than half a million.

Given that these are the same games represented in different tools, we can begin to see the magnitude of the accidental complexity.

Another way of looking at essential complexity is to consider how we explain the rules of the game to the player, which is in many cases a regulatory requirement. In some respects, the rules contain less than the essential complexity as we don't include the full layout of the reelstrips, but we do need to explain the game to a large extent and we can't have thousands of pages of explanation, nor could the player understand it if we did.

Slot Designer is beginning to approach the abstraction level of the essential complexity, and even more so when used in large scale game development through the use of game reusable libraries of code and game components.

Not all games are complex, and when designing a tool to make these complex games possible we don't want to make simple games more difficult. A guiding principle behind Slot Designer has always been, as computer scientist Alan Kay[4] so famously said, *'Simple things should be simple, complex things should be possible.'*

WORKING WITH COMPLEXITY

If we had a tool that could model the rules given to the player on the pay table[5] we'd expect it to get better every year, working faster and with more useful statistical analysis and reporting. This is conceivable as it's a very high-level description of the game with nothing about *how* the results are calculated.

The moment we say how the results are calculated we lose most of our ability to make improvements.

With simulation we can, in principle, come up with optimizations to make the code run faster, but serious optimization tends to be very difficult. You end up spending more time optimizing the code than just waiting for it to run, and when you make the next game the optimization no longer works, and you have to start again. In practice it's not very useful.

In Excel you can simplify models, for example calculating a single line of a multiline game as you "know" the RTP will be the same, but these techniques are fairly well established, and progress is slow. We can see the effect of increasing difficultly in using Excel as more games are being designed using simulation; it was very uncommon 20 years ago.

[4] https://en.wikiquote.org/wiki/Alan_Kay

[5] https://en.wikipedia.org/wiki/Pay_table

The inevitable improvements in PC hardware every year are helpful, but only to a small extent. Games are growing in complexity faster than PC's can keep up.

So, how do we execute the pay table? We can't, for several reasons.

- Its an imprecise specification with ambiguity that's not readily apparent until you try to do it. If two mathematicians modeled the same game, they would likely come up with different answers.
- There's critical information missing, for example reelstrips and probability tables.
- English is not well suited to this kind of task. It's very difficult for a computer to derive meaning from a human language. Note how software engineers don't program in English, nor do they use visual languages except for one or two rare exceptions.

Slot Designer sits in the middle between the pay table and the detailed instructions on how the results are calculated. It contains a precise specification of the game, without specifying in detail how to calculate it. Slot Designer uses a high-level game modelling language to describe games, and the tool itself can be improved without changing them. New mathematical techniques and optimizations will work on *all* games, and I anticipate there's going to be a lot more of these.

After 21 years this is the first major redesign of the Slot Designer language due to a fundamental new approach to game modelling. I don't foresee this happening again for a very long time.

7.3 PRODUCT VERIFICATION

It is commercially important that the mathematical game design is correct and that the final product faithfully implements it, especially in in real-money gaming.

The implementation of the game in the product code is (or should be) simulated to provide a final verification of the theoretical game design, and to check for software implementation errors. This type of simulation is generally much slower than any custom simulator the game designer might use due to the overhead of running platform code which has other requirements than just executing the simulation. None the less, it is essential to at least check for software bugs in the product.

The game is simulated to model a large number of plays and the RTP determined. The RTP should be close to the theoretical RTP calculated by the game designer. While it won't be exactly the same, as the test is a random simulation, a significant discrepancy outside the allowable margin indicates a problem. At this point a comprehensive breakdown of the simulated results to show the statistics for the various parts of the game is essential for quick diagnosis.

If the only information is that the simulated RTP is 90.1% and the theoretical result is 90.0%, it could take a very long time to work out what is causing the error, especially since modifying the game and rerunning large simulations is very time consuming. To make matters worse this typically occurs when the game is almost finished.

Slot Designer provides a verification tool called "Compare Server Play" to aid in verifying the product code and quickly diagnose errors. The product software is modified to write log files recording reel stopping positions and corresponding credits won, and a series of games is randomly played. The verification utility reads the log file and verifies it against the Slot Designer game model. When a difference between the two implementations is found, the reel stopping positions (essentially the symbols on the screen), and the two different win values are reported. From there it's easy to diagnose the problem. The full product code simulation should still be run, but by that point an entire class of potential errors has been eliminated.

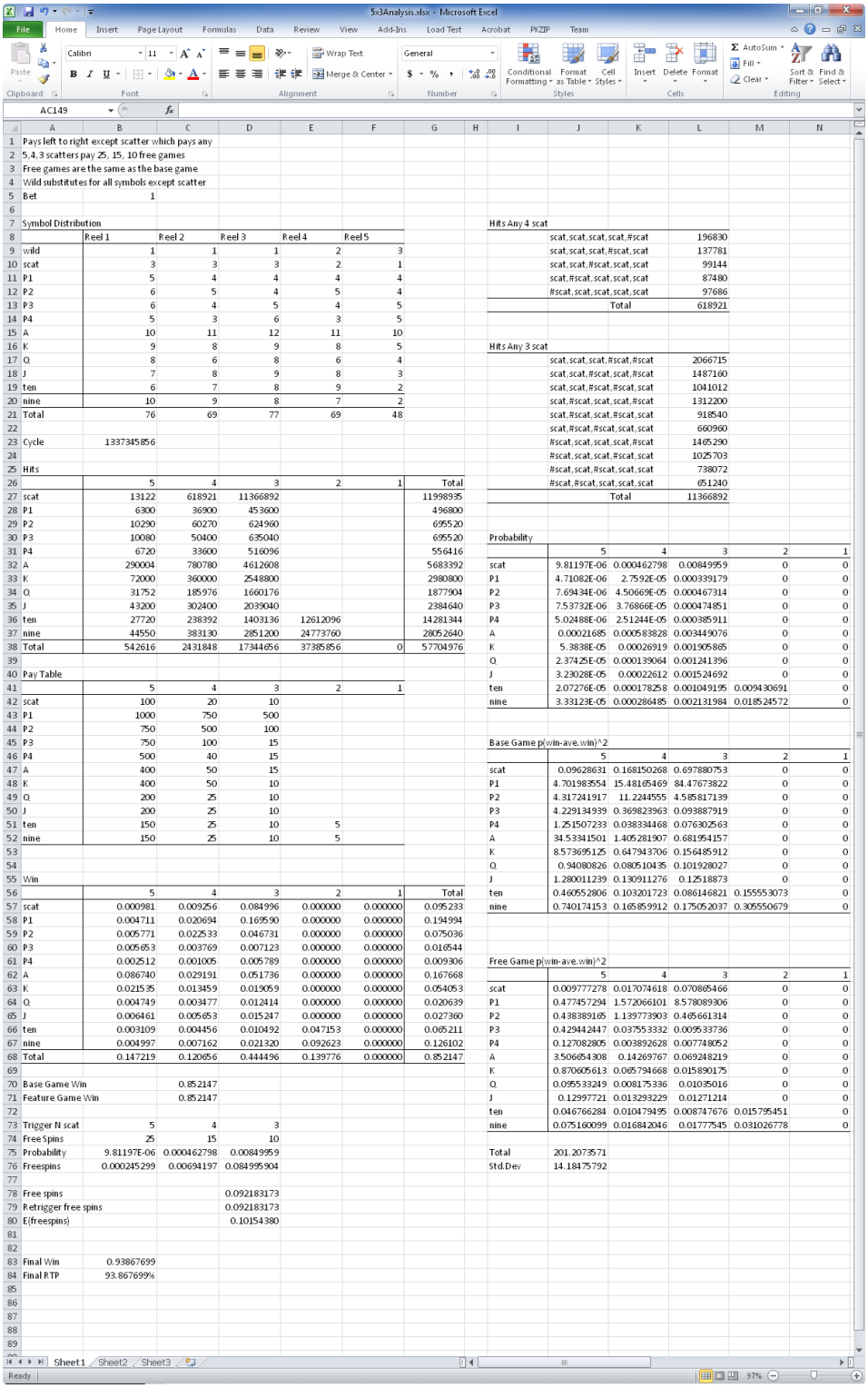
It's at this point another type of design error may appear. Excel game models cannot be played, and typically neither can simulators. So, while the game may be mathematically correct if this is the first chance the game designer has been able to play the game it might not be quite what they thought, and require rework to improve its performance. The psychology of game design is not apparent

though mathematical models and it is important. As with maths errors its costs a lot less if we can fix these issues earlier.

CHAPTER 8 ANALYSIS OF A GAME

To illustrate how the basic principles are used we calculate the RTP of a very simple base and free game, with left to right pays, wild, and scatter. Free games are triggered by 3, 4, or 5 scatter symbols and retrigger. First we use Excel, then Slot Designer using the SD3 language.

ANALYSIS USING MICROSOFT EXCEL



5x3Analysis.xlsx - Microsoft Excel

	A	B	C	D	E	F	G
1	Pays left to right except scatter which pays any						
2	5,4,3 scatters pay 25, 15, 10 free games						
3	Free games are the same as the base game						
4	Wild substitutes for all symbols except scatter						
5	Bet	1					
6							
7	Symbol Distribution						
8		Reel 1	Reel 2	Reel 3	Reel 4	Reel 5	
9	wild	1	1	1	2	3	
10	scat	3	3	3	2	1	
11	P1	5	4	4	4	4	
12	P2	6	5	4	5	4	
13	P3	6	4	5	4	5	
14	P4	5	3	6	3	5	
15	A	10	11	12	11	10	
16	K	9	8	9	8	5	
17	Q	8	6	8	6	4	
18	J	7	8	9	8	3	
19	ten	6	7	8	9	2	
20	nine	10	9	8	7	2	
21	Total	76	69	77	69	48	
22							
23	Cycle	1337345856					
24							
25	Hits						
26		5	4	3	2	1	Total
27	scat	13122	618921	11366892			11998935
28	P1	6300	36900	453600			496800
29	P2	10290	60270	624960			695520
30	P3	10080	50400	635040			695520
31	P4	6720	33600	516096			556416
32	A	290004	780780	4612608			5683392
33	K	72000	360000	2548800			2980800
34	Q	31752	185976	1660176			1877904
35	J	43200	302400	2039040			2384640
36	ten	27720	238392	1403136	12612096		14281344
37	nine	44550	383130	2851200	24773760		28052640
38	Total	542616	2431848	17344656	37385856	0	57704976
39							
40	Pay Table						
41		5	4	3	2	1	
42	scat	100	20	10			
43	P1	1000	750	500			
44	P2	750	500	100			
45	P3	750	100	15			
46	P4	500	40	15			
47	A	400	50	15			
48	K	400	50	10			
49	Q	200	25	10			
50	J	200	25	10			
51	ten	150	25	10	5		
52	nine	150	25	10	5		
53							
54							
55	Win						
56		5	4	3	2	1	Total
57	scat	0.000981	0.009256	0.084996	0.000000	0.000000	0.095233
58	P1	0.004711	0.020694	0.169590	0.000000	0.000000	0.194994
59	P2	0.005771	0.022533	0.046731	0.000000	0.000000	0.075036
60	P3	0.005653	0.003769	0.007123	0.000000	0.000000	0.016544
61	P4	0.002512	0.001005	0.005789	0.000000	0.000000	0.009306
62	A	0.086740	0.029191	0.051736	0.000000	0.000000	0.167668
63	K	0.021535	0.013459	0.019059	0.000000	0.000000	0.054053
64	Q	0.004749	0.003477	0.012414	0.000000	0.000000	0.020639
65	J	0.006461	0.005653	0.015247	0.000000	0.000000	0.027360
66	ten	0.003109	0.004456	0.010492	0.047153	0.000000	0.065211
67	nine	0.004997	0.007162	0.021320	0.092623	0.000000	0.126102
68	Total	0.147219	0.120656	0.444496	0.139776	0.000000	0.852147
69							
70	Base Game Win		0.852147				
71	Feature Game Win		0.852147				
72							
73	Trigger N scat	5	4	3			
74	Free Spins	25	15	10			
75	Probability	9.81197E-06	0.000462798	0.00849959			
76	Freespins	0.000245299	0.00694197	0.084995904			
77							
78	Free spins			0.092183173			
79	Retrigger free spins			0.092183173			
80	E(freespins)			0.10154380			
81							
82							
83	Final Win	0.93867699					
84	Final RTP	93.867699%					
85							
86							
87							
88							
89							

	I	J	K	L	M	N
	Hits Any 4 scat					
		scat,scat,scat,scat,#scat		196830		
		scat,scat,scat,#scat,scat		137781		
		scat,scat,#scat,scat,scat		99144		
		scat,#scat,scat,scat,scat		87480		
		#scat,scat,scat,scat,scat		97686		
			Total	618921		
	Hits Any 3 scat					
		scat,scat,scat,#scat,#scat		2066715		
		scat,scat,#scat,scat,#scat		1487160		
		scat,scat,#scat,#scat,scat		1041012		
		scat,#scat,scat,scat,#scat		1312200		
		scat,#scat,scat,#scat,scat		918540		
		scat,#scat,#scat,scat,scat		660960		
		#scat,scat,scat,scat,#scat		1465290		
		#scat,scat,scat,#scat,scat		1025703		
		#scat,scat,#scat,scat,scat		738072		
		#scat,#scat,scat,scat,scat		651240		
			Total	11366892		
	Probability					
		5	4	3	2	1
	scat	9.81197E-06	0.000462798	0.00849959	0	0
	P1	4.71082E-06	2.7592E-05	0.000339179	0	0
	P2	7.69434E-06	4.50669E-05	0.000467314	0	0
	P3	7.53732E-06	3.76866E-05	0.000474851	0	0
	P4	5.02488E-06	2.51244E-05	0.000385911	0	0
	A	0.00021685	0.000583828	0.003449076	0	0
	K	5.3838E-05	0.00026919	0.001905865	0	0
	Q	2.37425E-05	0.000139064	0.001241396	0	0
	J	3.23028E-05	0.00022612	0.001524692	0	0
	ten	2.07276E-05	0.000178258	0.001049195	0.009430691	0
	nine	3.33123E-05	0.000286485	0.002131984	0.018524572	0
	Base Game p(win-ave.win)^2					
		5	4	3	2	1
	scat	0.09628631	0.168150268	0.697880753	0	0
	P1	4.701983554	15.48165469	84.47673822	0	0
	P2	4.317241917	11.2244555	4.585817139	0	0
	P3	4.229134939	0.369823963	0.093887919	0	0
	P4	1.251507233	0.038334468	0.076302563	0	0
	A	34.53341501	1.405281907	0.681954157	0	0
	K	8.573695125	0.647943706	0.156485912	0	0
	Q	0.94080826	0.080510435	0.101928027	0	0
	J	1.280011239	0.130911276	0.12518873	0	0
	ten	0.460552806	0.103201723	0.086146821	0.155553073	0
	nine	0.740174153	0.165859912	0.175052037	0.305550679	0
	Free Game p(win-ave.win)^2					
		5	4	3	2	1
	scat	0.009777278	0.017074618	0.070865466	0	0
	P1	0.477457294	1.572066101	8.578089306	0	0
	P2	0.438389165	1.139773903	0.465661314	0	0
	P3	0.429442447	0.037553332	0.009533736	0	0
	P4	0.127082805	0.003892628	0.007748052	0	0
	A	3.506654308	0.14269767	0.069248219	0	0
	K	0.870605613	0.065794668	0.015890175	0	0
	Q	0.095533249	0.008175336	0.01035016	0	0
	J	0.12997721	0.013293229	0.01271214	0	0
	ten	0.046766284	0.010479495	0.008747676	0.015795451	0
	nine	0.075160099	0.016842046	0.01777545	0.031026778	0
	Total	201.2073571				
	Std.Dev	14.18475792				

The calculations are illustrated by calculating several exemplary values from which the spreadsheet can be understood.

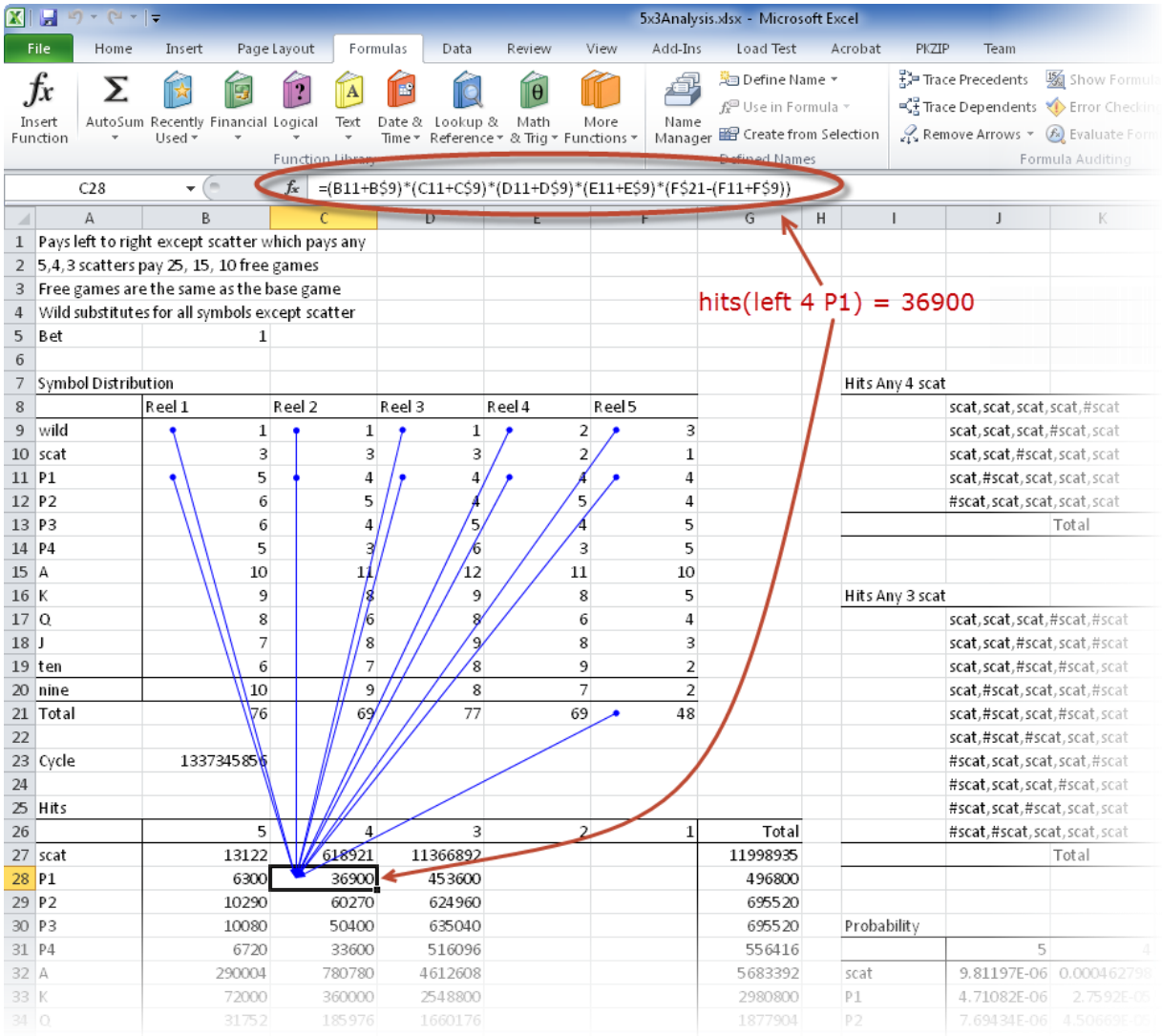
First we calculate the hits for the payline combinations and the scatters to determine the average base and free game win. We then determine the probability of the free games occurring and combine the base and free games wins to calculate the final RTP of the game. Last, we calculate the standard deviation.

PAYLINE COMBINATIONS HITS

The calculation of hits for left 4 P1 is shown, and includes the wild symbols.

$$\begin{aligned} hits(left\ 4\ P1) &= hits(P1, P1, P1, P1, \#P1) \\ &= (5+1)\times(4+1)\times(4+1)\times(4+2)\times(48-4-3) \\ &= 36900 \end{aligned}$$ Equation 37

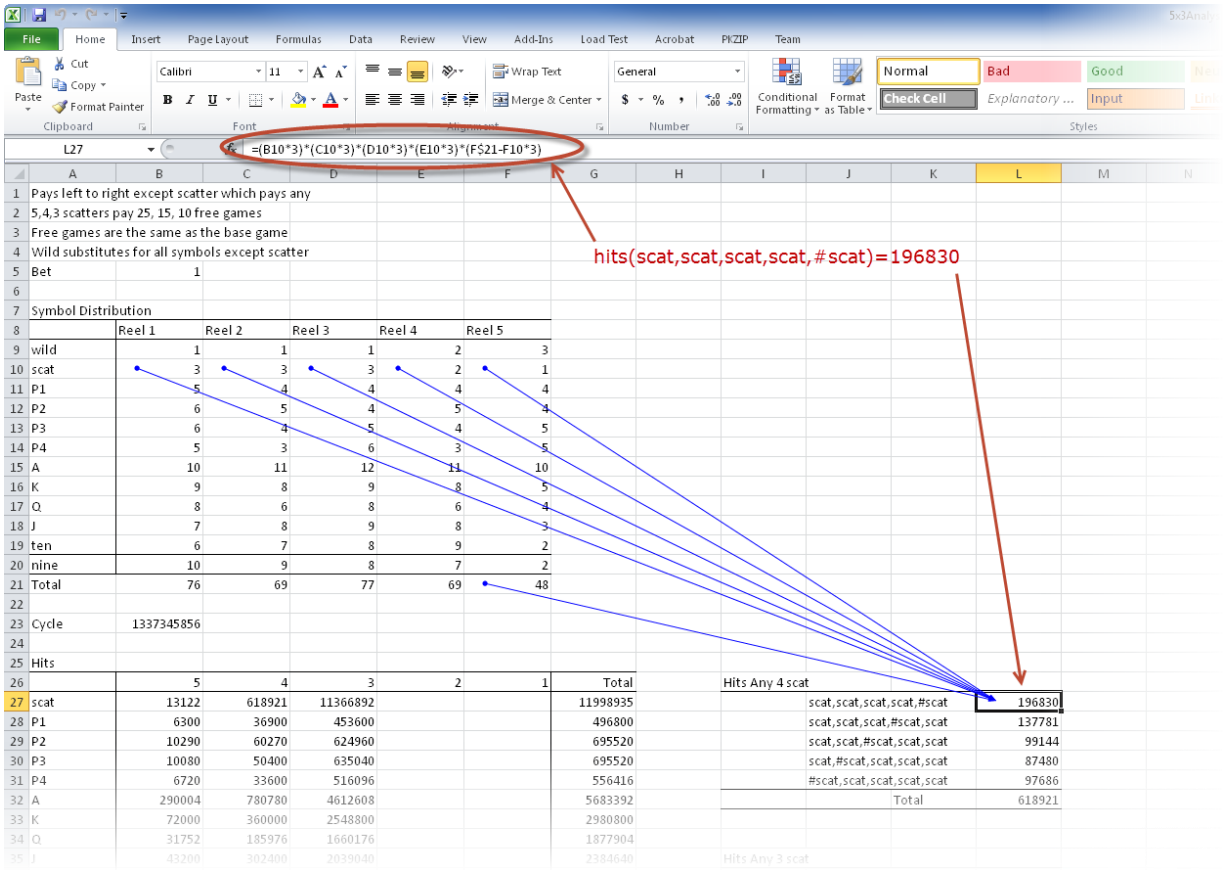
The other non-scatter hits are calculated in a similar manner.



SCATTER COMBINATION HITS

Scatter hits are a little more complex as they are "any" pay rules rather than left to right, and are the sum of the different combinations that make up the rule. For the rule "any 4 scat" we need to calculate the hits for 5 combinations, the first of which is

$$\begin{aligned} hits(scat, scat, scat, scat, \#scat) &= (3*3)\times(3*3)\times(3*3)\times(2*3)\times(48-1*3) \\ &= 196830 \end{aligned} \qquad \text{Equation 38}$$



	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Pays left to right except scatter which pays any												
2	5,4,3 scatters pay 25, 15, 10 free games												
3	Free games are the same as the base game												
4	Wild substitutes for all symbols except scatter												
5	Bet	1											
6													
7	Symbol Distribution												
8		Reel 1	Reel 2	Reel 3	Reel 4	Reel 5							
9	wild	1	1	1	2	3							
10	scat	3	3	3	2	1							
11	P1	5	4	4	4	4							
12	P2	6	5	4	5	4							
13	P3	6	4	5	4	5							
14	P4	5	3	6	3	5							
15	A	10	11	12	11	10							
16	K	9	8	9	8	5							
17	Q	8	6	8	6	4							
18	J	7	8	9	8	3							
19	ten	6	7	8	9	2							
20	nine	10	9	8	7	2							
21	Total	76	69	77	69	48							
22													
23	Cycle	1337345856											
24													
25	Hits												
26		5	4	3	2	1	Total		Hits Any 4 scat				
27	scat	13122	618921	11366892			11998935			scat,scat,scat,scat,#scat		196830	
28	P1	6300	36900	453600			496800			scat,scat,scat,#scat,scat		137781	
29	P2	10290	60270	624960			695520			scat,scat,#scat,scat,scat		99144	
30	P3	10080	50400	635040			695520			scat,#scat,scat,scat,scat		87480	
31	P4	6720	33600	516096			556416			#scat,scat,scat,scat,scat		97686	
32	A	290004	780780	4612608			5683392				Total	618921	
33	K	72000	360000	2548800			2980800						
34	Q	31752	185976	1660176			1877904						
35	J	43200	302400	2039040			2384640		Hits Any 3 scat				

AVERAGE WIN

From the hits we can calculate the probability of each rule, and from that the average credit win.

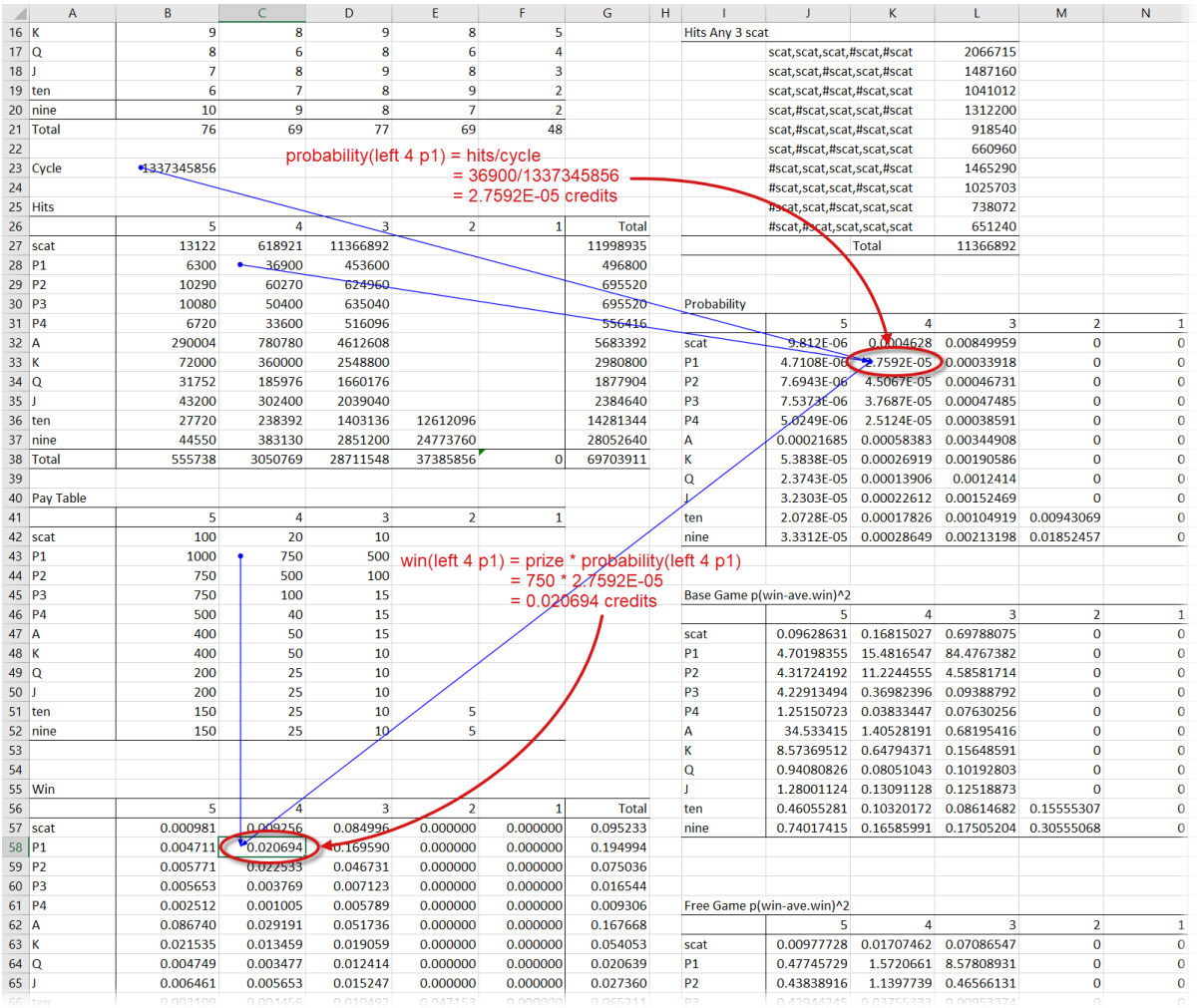
The probability of "left 4 P1" is 2.7592×10^{-5}.

$$\begin{aligned} probability(left\ 4\ P1) &= \frac{hits(left\ 4\ P1)}{cycle} \\ &= \frac{36900}{1337345856} \\ &= 2.7592\cdot10^{-5} \end{aligned} \qquad \text{Equation 39}$$

The average win for "left 4 P1" is then 0.020694 credits.

$$\begin{aligned} win(left\ 4\ P1) &= probability \times prize \\ &= 2.7592\cdot10^{-5}\times750 \\ &= 0.20694 \end{aligned} \qquad \text{Equation 40}$$

The total win of the base game is the sum of the wins of all the rules in the pay table. As the free game has identical rules to the base game it has the same average win.



	A	B	C	D	E	F	G
16	K	9	8	9	8	5	
17	Q	8	6	8	6	4	
18	J	7	8	9	8	3	
19	ten	6	7	8	9	2	
20	nine	10	9	8	7	2	
21	Total	76	69	77	69	48	
22							
23	Cycle	1337345856					
24							
25	Hits						
26		5	4	3	2	1	Total
27	scat	13122	618921	11366892			11998935
28	P1	6300	36900	453600			496800
29	P2	10290	60270	624960			695520
30	P3	10080	50400	635040			695520
31	P4	6720	33600	516096			556416
32	A	290004	780780	4612608			5683392
33	K	72000	360000	2548800			2980800
34	Q	31752	185976	1660176			1877904
35	J	43200	302400	2039040			2384640
36	ten	27720	238392	1403136	12612096		14281344
37	nine	44550	383130	2851200	24773760		28052640
38	Total	555738	3050769	28711548	37385856	0	69703911
39							
40	Pay Table						
41		5	4	3	2	1	
42	scat	100	20	10			
43	P1	1000	750	500			
44	P2	750	500	100			
45	P3	750	100	15			
46	P4	500	40	15			
47	A	400	50	15			
48	K	400	50	10			
49	Q	200	25	10			
50	J	200	25	10			
51	ten	150	25	10	5		
52	nine	150	25	10	5		
53							
54							
55	Win						
56		5	4	3	2	1	Total
57	scat	0.000981	0.009256	0.084996	0.000000	0.000000	0.095233
58	P1	0.004711	0.020694	0.169590	0.000000	0.000000	0.194994
59	P2	0.005771	0.022533	0.046731	0.000000	0.000000	0.075036
60	P3	0.005653	0.003769	0.007123	0.000000	0.000000	0.016544
61	P4	0.002512	0.001005	0.005789	0.000000	0.000000	0.009306
62	A	0.086740	0.029191	0.051736	0.000000	0.000000	0.167668
63	K	0.021535	0.013459	0.019059	0.000000	0.000000	0.054053
64	Q	0.004749	0.003477	0.012414	0.000000	0.000000	0.020639
65	J	0.006461	0.005653	0.015247	0.000000	0.000000	0.027360

	I	J	K	L	M	N
	Hits Any 3 scat					
		scat,scat,scat,#scat,#scat		2066715		
		scat,scat,#scat,scat,#scat		1487160		
		scat,scat,#scat,#scat,scat		1041012		
		scat,#scat,scat,scat,#scat		1312200		
		scat,#scat,scat,#scat,scat		918540		
		scat,#scat,#scat,scat,scat		660960		
		#scat,scat,scat,scat,#scat		1465290		
		#scat,scat,scat,#scat,scat		1025703		
		#scat,scat,#scat,scat,scat		738072		
		#scat,#scat,scat,scat,scat		651240		
			Total	11366892		
	Probability					
		5	4	3	2	1
	scat	9.812E-06	0.0004628	0.00849959	0	0
	P1	4.7108E-06	2.7592E-05	0.00033918	0	0
	P2	7.6943E-06	4.5067E-05	0.00046731	0	0
	P3	7.5373E-06	3.7687E-05	0.00047485	0	0
	P4	5.0249E-06	2.5124E-05	0.00038591	0	0
	A	0.00021685	0.00058383	0.00344908	0	0
	K	5.3838E-05	0.00026919	0.00190586	0	0
	Q	2.3743E-05	0.00013906	0.0012414	0	0
	J	3.2303E-05	0.00022612	0.00152469	0	0
	ten	2.0728E-05	0.00017826	0.00104919	0.00943069	0
	nine	3.3312E-05	0.00028649	0.00213198	0.01852457	0
	Base Game p(win-ave.win)^2					
		5	4	3	2	1
	scat	0.09628631	0.16815027	0.69788075	0	0
	P1	4.70198355	15.4816547	84.4767382	0	0
	P2	4.31724192	11.2244555	4.58581714	0	0
	P3	4.22913494	0.36982396	0.09388792	0	0
	P4	1.25150723	0.03833447	0.07630256	0	0
	A	34.533415	1.40528191	0.68195416	0	0
	K	8.57369512	0.64794371	0.15648591	0	0
	Q	0.94080826	0.08051043	0.10192803	0	0
	J	1.28001124	0.13091128	0.12518873	0	0
	ten	0.46055281	0.10320172	0.08614682	0.15555307	0
	nine	0.74017415	0.16585991	0.17505204	0.30555068	0
	Free Game p(win-ave.win)^2					
		5	4	3	2	1
	scat	0.00977728	0.01707462	0.07086547	0	0
	P1	0.47745729	1.5720661	8.57808931	0	0
	P2	0.43838916	1.1397739	0.46566131	0	0

COMBINE BASE AND FREE GAME WIN TO CALCULATE FINAL RTP

Using Equation 28 (page 15) we calculate the final RTP. First, the expected number of free spins triggered, per spin, by the base game is is the sum of the individual triggers, 0.092183173 (cell D78)

$$\begin{aligned} freespins &= \sum n \cdot p \\ &= 25 \times 9.81197 \cdot 10^{-6} + 15 \times 0.000462798 + 10 \times 0.00849959 \\ &= 0.092183173 \end{aligned}$$

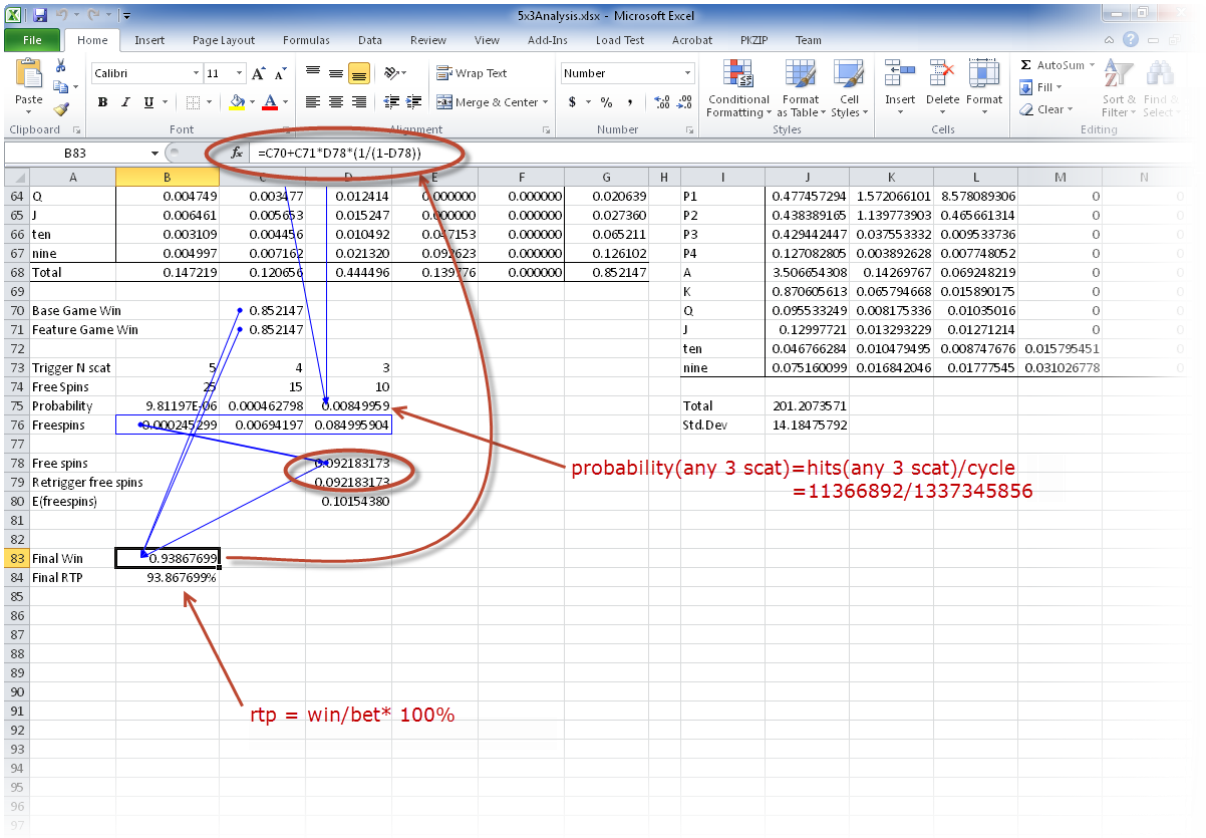
Equation 41

The number of free spins (re)triggered by the free game is the same as the base game, as the games are identical (except for the bet), and the free game retriggers.

The final RTP (cell B84) is then 93.867699%

$$\begin{aligned} F &= \frac{B_0 + B_1 n_1 p_1 \left(\frac{1}{1 - n_2 p_2} \right)}{bet} \times 100\% \\ &= \frac{0.852147 + 0.852147 \times 0.092183173 \left(\frac{1}{1 - 0.092183173} \right)}{1} \times 100\% \\ &= 93.867699\% \end{aligned}$$

Equation 42



	A	B	C	D	E	F	G	H	I	J	K	L	M	N
64	Q	0.004749	0.003477	0.012414	0.000000	0.000000	0.020639		P1	0.477457294	1.572066101	8.578089306	0	0
65	J	0.006461	0.005653	0.015247	0.000000	0.000000	0.027360		P2	0.438389165	1.139773903	0.465661314	0	0
66	ten	0.003109	0.004456	0.010492	0.017153	0.000000	0.065211		P3	0.429442447	0.037553332	0.009533736	0	0
67	nine	0.004997	0.007162	0.021320	0.092623	0.000000	0.126102		P4	0.127082805	0.003892628	0.007748052	0	0
68	Total	0.147219	0.120656	0.444496	0.139776	0.000000	0.852147		A	3.506654308	0.14269767	0.069248219	0	0
69									K	0.870605613	0.065794668	0.015890175	0	0
70	Base Game Win		0.852147						Q	0.095533249	0.008175336	0.01035016	0	0
71	Feature Game Win		0.852147						J	0.12997721	0.013293229	0.01271214	0	0
72									ten	0.046766284	0.010479495	0.008747676	0.015795451	0
73	Trigger N scat	5	4	3					nine	0.075160099	0.016842046	0.01777545	0.031026778	0
74	Free Spins	25	15	10										
75	Probability	9.81197E-06	0.000462798	0.00849959					Total	201.2073571				
76	Freespins	0.000245299	0.00694197	0.084995904					Std.Dev	14.18475792				
77														
78	Free spins			0.092183173										
79	Retrigger free spins			0.092183173										
80	E(freespins)			0.10154380										
81														
82														
83	Final Win	0.93867699												
84	Final RTP	93.867699%												

STANDARD DEVIATION

We calculate the standard deviation using Equation 29 (page 15). We build two tables for containing the probability of a prize occuring times the square of the prize minus the games average win. Each cell contains

$$p_i\left(x_i-\bar{x}\right)^2$$ Equation 43

The probability for base game prizes is simply the probability of the combination occurring, which we calculated as 2.7592×10^{-5} for left 4 P1 with Equation 39 (page 33). Thus, the cell (K48) representing "left 4 P1" contains

$$2.7592\cdot10^{-5}\left(750-0.93867699\right)^2=15.48165469$$ Equation 44

The corresponding cell in the free game uses takes into account the probability of free game occurring (after the initial trigger and any number of retriggers), which is the expected number of free spins from Equation 27 (page 15), and as shown in the similar Equation 42, is 0.10154380 (cell D80).

$$\begin{aligned}E(\,freespins)&=n_1p_1\left(\frac{1}{1-n_2p_2}\right)\\&=0.092183173\left(\frac{1}{1-0.092183173}\right)\\&=0.10154380\end{aligned}$$ Equation 45

The table entry for left 4 P1 in the free game is then 1.572066101 (cell K64)

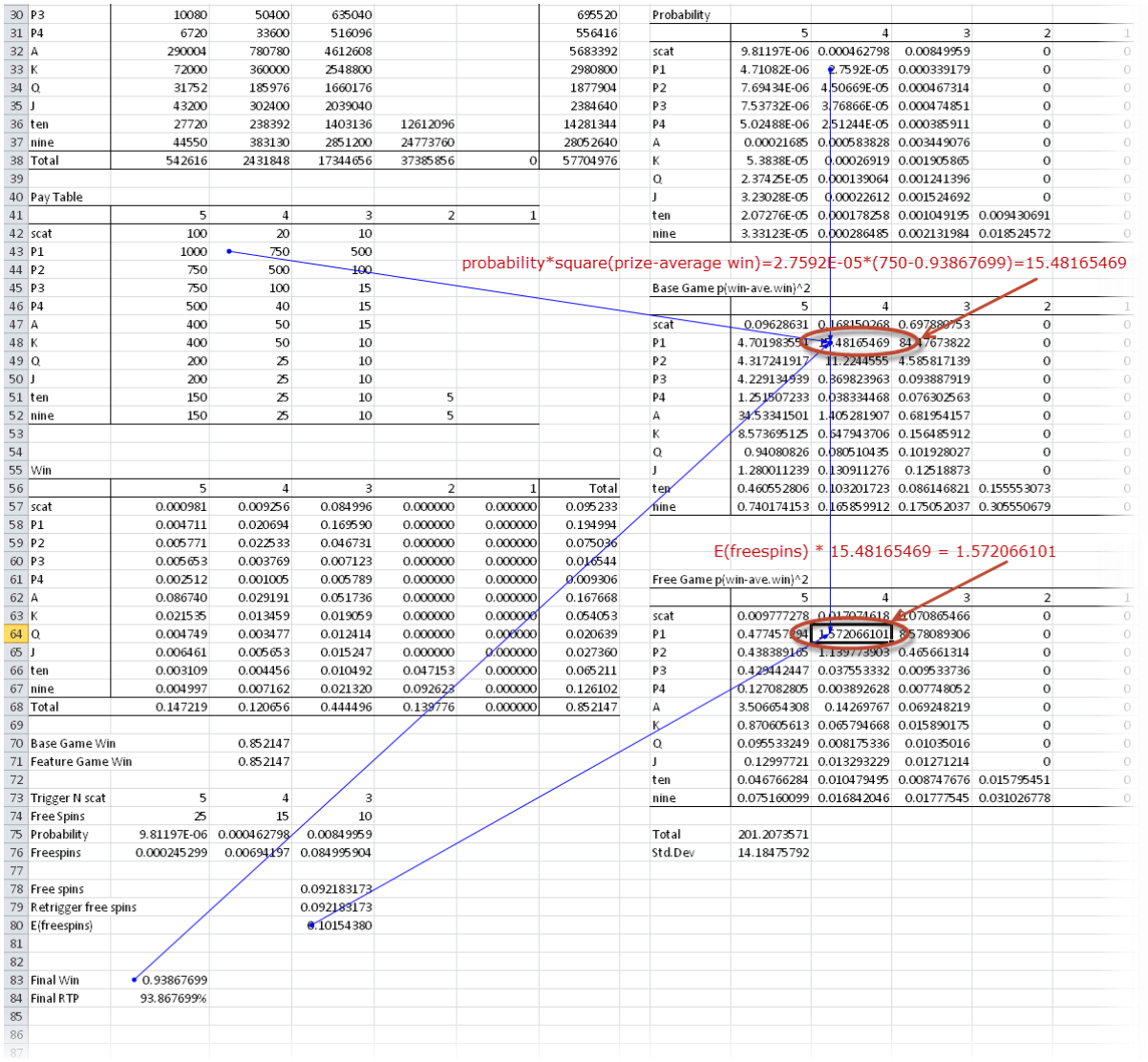
$$0.10154380\times15.48165469=1.572066101$$ Equation 46

The standard deviation (cell J76) is then the square root of the sum of these two tables

$$\sigma = \sqrt{201.2073571}$$
$$= 14.18475792$$

Equation 47

The relevant part of the spread sheet in more detail



P3	10080	50400	635040			695520
P4	6720	33600	516096			556416
A	290004	780780	4612608			5683392
K	72000	360000	2548800			2980800
Q	31752	185976	1660176			1877904
J	43200	302400	2039040			2384640
ten	27720	238392	1403136	12612096		14281344
nine	44550	383130	2851200	24773760		28052640
Total	542616	2431848	17344656	37385856	0	57704976

Pay Table					
	5	4	3	2	1
scat	100	20	10		
P1	1000	750	500		
P2	750	500	100		
P3	750	100	15		
P4	500	40	15		
A	400	50	15		
K	400	50	10		
Q	200	25	10		
J	200	25	10		
ten	150	25	10	5	
nine	150	25	10	5	

Win						
	5	4	3	2	1	Total
scat	0.000981	0.009256	0.084996	0.000000	0.000000	0.095233
P1	0.004711	0.020694	0.169590	0.000000	0.000000	0.194994
P2	0.005771	0.022533	0.046731	0.000000	0.000000	0.075036
P3	0.005653	0.003769	0.007123	0.000000	0.000000	0.016544
P4	0.002512	0.001005	0.005789	0.000000	0.000000	0.009306
A	0.086740	0.029191	0.051736	0.000000	0.000000	0.167668
K	0.021535	0.013459	0.019059	0.000000	0.000000	0.054053
Q	0.004749	0.003477	0.012414	0.000000	0.000000	0.020639
J	0.006461	0.005653	0.015247	0.000000	0.000000	0.027360
ten	0.003109	0.004456	0.010492	0.047153	0.000000	0.065211
nine	0.004997	0.007162	0.021320	0.092623	0.000000	0.126102
Total	0.147219	0.120656	0.444496	0.139776	0.000000	0.852147

Base Game Win		0.852147	
Feature Game Win		0.852147	
Trigger N scat	5	4	3
Free Spins	25	15	10
Probability	9.81197E-06	0.000462798	0.00849959
Freespins	0.000245299	0.00694197	0.084995904
Free spins			0.092183173
Retrigger free spins			0.092183173
E(freespins)			0.10154380
Final Win	0.93867699		
Final RTP	93.867699%		

Probability					
	5	4	3	2	1
scat	9.81197E-06	0.000462798	0.00849959	0	0
P1	4.71082E-06	2.7592E-05	0.000339179	0	0
P2	7.69434E-06	4.50669E-05	0.000467314	0	0
P3	7.53732E-06	3.76866E-05	0.000474851	0	0
P4	5.02488E-06	2.51244E-05	0.000385911	0	0
A	0.00021685	0.000583828	0.003449076	0	0
K	5.3838E-05	0.00026919	0.001905865	0	0
Q	2.37425E-05	0.000139064	0.001241396	0	0
J	3.23028E-05	0.00022612	0.001524692	0	0
ten	2.07276E-05	0.000178258	0.001049195	0.009430691	0
nine	3.33123E-05	0.000286485	0.002131984	0.018524572	0

probability*square(prize-average win)=2.7592E-05*(750-0.93867699)=15.48165469

Base Game p(win-ave.win)^2					
	5	4	3	2	1
scat	0.09628631	0.168150268	0.697880753	0	0
P1	4.701983591	15.48165469	84.47673822	0	0
P2	4.317241917	11.2244555	4.585817139	0	0
P3	4.229134939	0.369823963	0.093887919	0	0
P4	1.251507233	0.038334468	0.076302563	0	0
A	34.53341501	1.405281907	0.681954157	0	0
K	8.573695125	0.647943706	0.156485912	0	0
Q	0.94080826	0.080510435	0.101928027	0	0
J	1.280011239	0.130911276	0.12518873	0	0
ten	0.460552806	0.103201723	0.086146821	0.155553073	0
nine	0.740174153	0.165859912	0.175052037	0.305550679	0

E(freespins) * 15.48165469 = 1.572066101

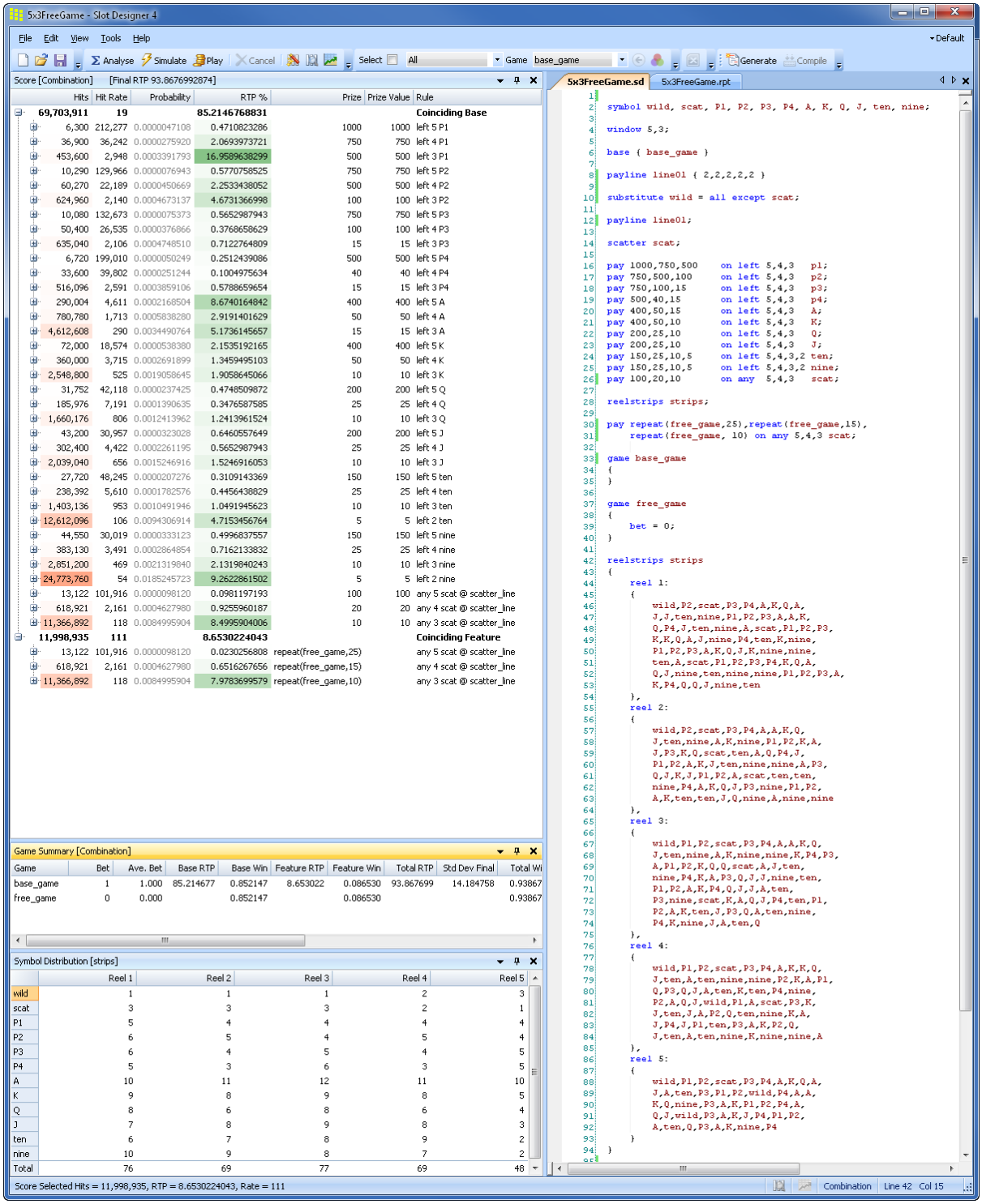
Free Game p(win-ave.win)^2					
	5	4	3	2	1
scat	0.009777278	0.017074618	0.070865466	0	0
P1	0.477457894	1.572066101	8.578089306	0	0
P2	0.438389165	1.139773903	0.465661314	0	0
P3	0.429142447	0.037553332	0.009533736	0	0
P4	0.127082805	0.003892628	0.007748052	0	0
A	3.506654308	0.14269767	0.069248219	0	0
K	0.870605613	0.065794668	0.015890175	0	0
Q	0.095533249	0.008175336	0.01035016	0	0
J	0.12997721	0.013293229	0.01271214	0	0
ten	0.046766284	0.010479495	0.008747676	0.015795451	0
nine	0.075160099	0.016842046	0.01777545	0.031026778	0

Total	201.2073571
Std.Dev	14.18475792

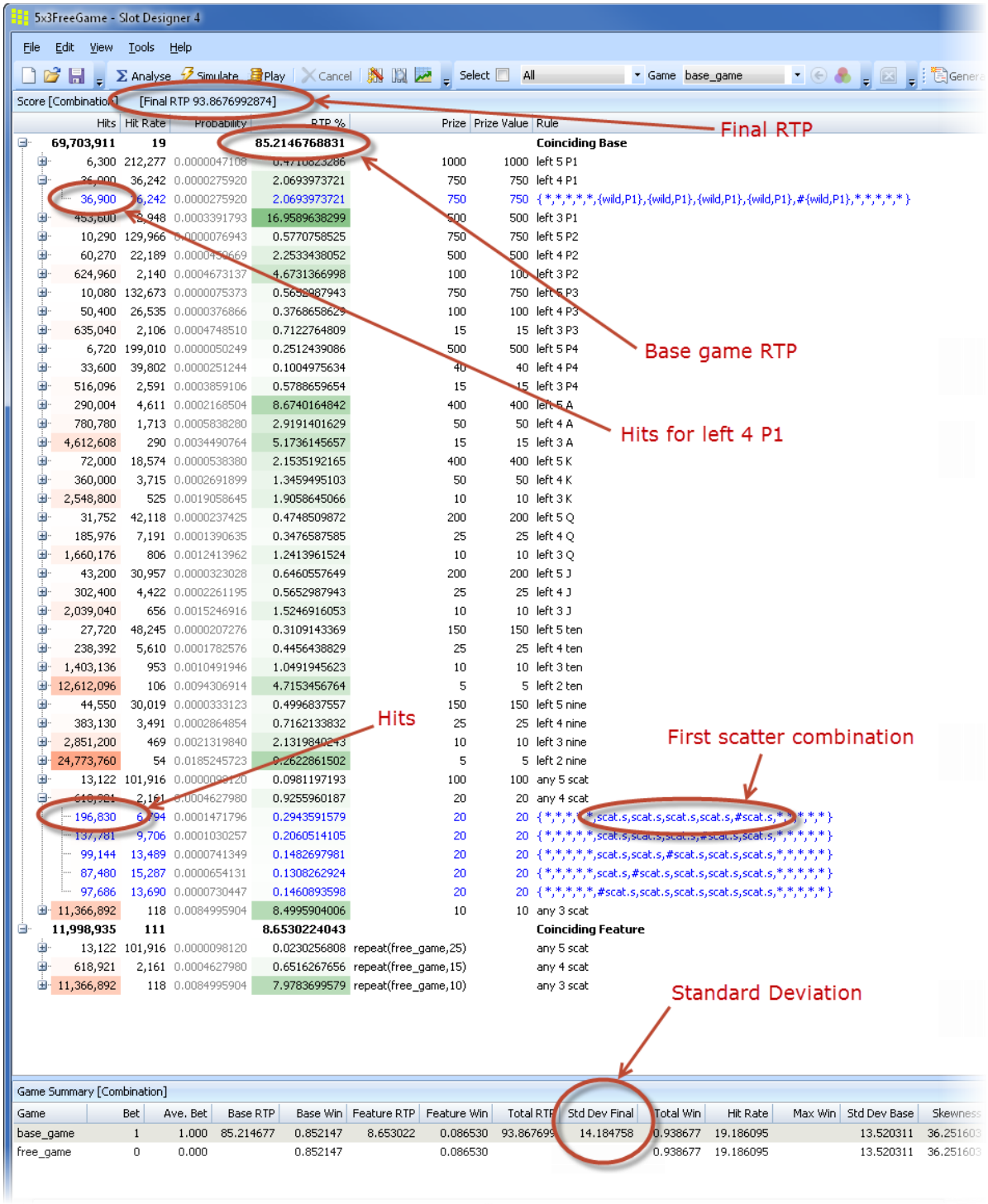
As noted previously the standard deviation here does not include the zero wins which is inaccurate, but we cannot accurately calculate it due to coinciding wins.

Analysis Using Slot Designer

The same is shown in the Slot Designer IDE. Unlike in Excel where the calculations are specified, Slot Designer works at a higher level of abstraction and performs whatever calculations are required itself. The 'code' describing the game shown on the right, and the calculated game statistics on the left.

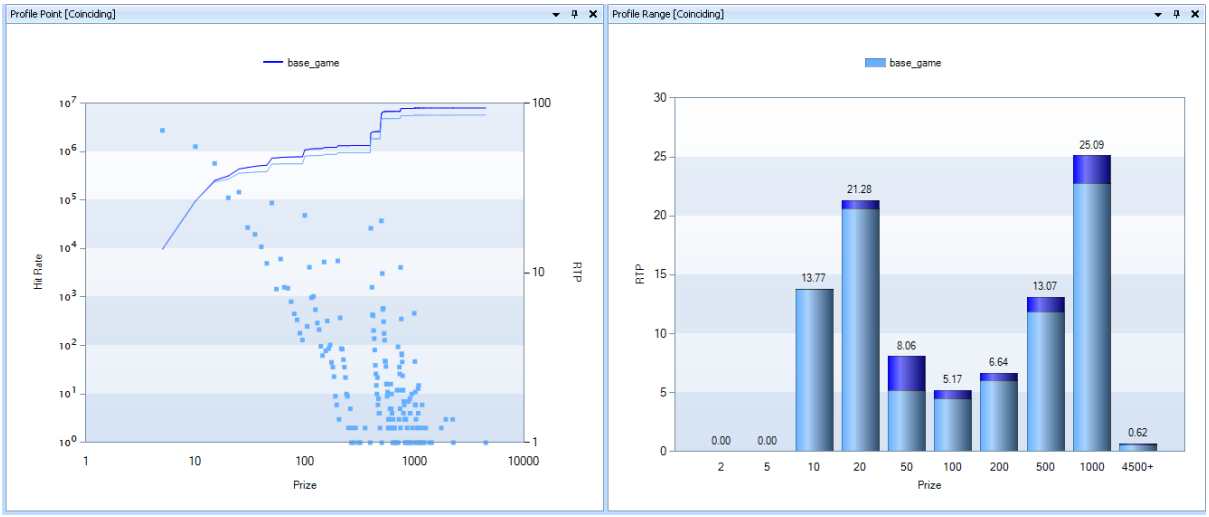

5x3FreeGame - Slot Designer 4
File Edit View Tools Help
Default
Analyse Simulate Play Cancel Select All Game base_game Generate Compile
Score [Combination] [Final RTP 93.8676992874]
Hits Hit Rate Probability RTP % Prize Prize Value Rule
69,703,911 19 85.2146768831 Coinciding Base
6,300 212,277 0.0000047108 0.4710823286 1000 1000 left 5 P1
36,900 36,242 0.0000275920 2.0693973721 750 750 left 4 P1
453,600 2,948 0.0003391793 16.9589638299 500 500 left 3 P1
10,290 129,966 0.0000076943 0.5770758525 750 750 left 5 P2
60,270 22,189 0.0000450669 2.2533438052 500 500 left 4 P2
624,960 2,140 0.0004673137 4.6731366998 100 100 left 3 P2
10,080 132,673 0.0000075373 0.5652987943 750 750 left 5 P3
50,400 26,535 0.0000376866 0.3768658629 100 100 left 4 P3
635,040 2,106 0.0004748510 0.7122764809 15 15 left 3 P3
6,720 199,010 0.0000050249 0.2512439086 500 500 left 5 P4
33,600 39,802 0.0000251244 0.1004975634 40 40 left 4 P4
516,096 2,591 0.0003859106 0.5788659654 15 15 left 3 P4
290,004 4,611 0.0002168504 8.6740164842 400 400 left 5 A
780,780 1,713 0.0005838280 2.9191401629 50 50 left 4 A
4,612,608 290 0.0034490764 5.1736145657 15 15 left 3 A
72,000 18,574 0.0000538380 2.1535192165 400 400 left 5 K
360,000 3,715 0.0002691899 1.3459495103 50 50 left 4 K
2,548,800 525 0.0019058645 1.9058645066 10 10 left 3 K
31,752 42,118 0.0000237425 0.4748509872 200 200 left 5 Q
185,976 7,191 0.0001390635 0.3476587585 25 25 left 4 Q
1,660,176 806 0.0012413962 1.2413961524 10 10 left 3 Q
43,200 30,957 0.0000323028 0.6460557649 200 200 left 5 J
302,400 4,422 0.0002261195 0.5652987943 25 25 left 4 J
2,039,040 656 0.0015246916 1.5246916053 10 10 left 3 J
27,720 48,245 0.0000207276 0.3109143369 150 150 left 5 ten
238,392 5,610 0.0001782576 0.4456438829 25 25 left 4 ten
1,403,136 953 0.0010491946 1.0491945623 10 10 left 3 ten
12,612,096 106 0.0094306914 4.7153456764 5 5 left 2 ten
44,550 30,019 0.0000333123 0.4996837557 150 150 left 5 nine
383,130 3,491 0.0002864854 0.7162133832 25 25 left 4 nine
2,851,200 469 0.0021319840 2.1319840243 10 10 left 3 nine
24,773,760 54 0.0185245723 9.2622861502 5 5 left 2 nine
13,122 101,916 0.0000098120 0.0981197193 100 100 any 5 scat @ scatter_line
618,921 2,161 0.0004627980 0.9255960187 20 20 any 4 scat @ scatter_line
11,366,892 118 0.0084995904 8.4995904006 10 10 any 3 scat @ scatter_line
11,998,935 111 8.6530224043 Coinciding Feature
13,122 101,916 0.0000098120 0.0230256808 repeat(free_game,25) any 5 scat @ scatter_line
618,921 2,161 0.0004627980 0.6516267656 repeat(free_game,15) any 4 scat @ scatter_line
11,366,892 118 0.0084995904 7.9783699579 repeat(free_game,10) any 3 scat @ scatter_line
Game Summary [Combination]
Game Bet Ave. Bet Base RTP Base Win Feature RTP Feature Win Total RTP Std Dev Final Total Wi
base_game 1 1.000 85.214677 0.852147 8.653022 0.086530 93.867699 14.184758 0.93867
free_game 0 0.000 0.852147 0.086530 0.93867
Symbol Distribution [strips]
Reel 1 Reel 2 Reel 3 Reel 4 Reel 5
wild 1 1 1 2 3
scat 3 3 3 2 1
P1 5 4 4 4 4
P2 6 5 4 5 4
P3 6 4 5 4 5
P4 5 3 6 3 5
A 10 11 12 11 10
K 9 8 9 8 5
Q 8 6 8 6 4
J 7 8 9 8 3
ten 6 7 8 9 2
nine 10 9 8 7 2
Total 76 69 77 69 48
5x3FreeGame.sd
5x3FreeGame.rpt
symbol wild, scat, P1, P2, P3, P4, A, K, Q, J, ten, nine;
window 5,3;
base { base_game }
payline line01 { 2,2,2,2,2 }
substitute wild = all except scat;
payline line01;
scatter scat;
pay 1000,750,500 on left 5,4,3 p1;
pay 750,500,100 on left 5,4,3 p2;
pay 750,100,15 on left 5,4,3 p3;
pay 500,40,15 on left 5,4,3 p4;
pay 400,50,15 on left 5,4,3 A;
pay 400,50,10 on left 5,4,3 K;
pay 200,25,10 on left 5,4,3 Q;
pay 200,25,10 on left 5,4,3 J;
pay 150,25,10,5 on left 5,4,3,2 ten;
pay 150,25,10,5 on left 5,4,3,2 nine;
pay 100,20,10 on any 5,4,3 scat;
reelstrips strips;
pay repeat(free_game,25),repeat(free_game,15),
repeat(free_game, 10) on any 5,4,3 scat;
game base_game
{
}
game free_game
{
bet = 0;
}
reelstrips strips
{
reel 1:
{
wild,P2,scat,P3,P4,A,K,Q,A,
J,J,ten,nine,P1,P2,P3,A,A,K,
Q,P4,J,ten,nine,A,scat,P1,P2,P3,
K,K,Q,A,J,nine,P4,ten,K,nine,
P1,P2,P3,A,K,Q,J,K,nine,nine,
ten,A,scat,P1,P2,P3,P4,K,Q,A,
Q,J,nine,ten,nine,nine,P1,P2,P3,A,
K,P4,Q,Q,J,nine,ten
},
reel 2:
{
wild,P2,scat,P3,P4,A,A,K,Q,
J,ten,nine,A,K,nine,P1,P2,K,A,
J,P3,K,Q,scat,ten,A,Q,P4,J,
P1,P2,A,K,J,ten,nine,nine,A,P3,
Q,J,K,J,P1,P2,A,scat,ten,ten,
nine,P4,A,K,Q,J,P3,nine,P1,P2,
A,K,ten,ten,J,Q,nine,A,nine,nine
},
reel 3:
{
wild,P1,P2,scat,P3,P4,A,A,K,Q,
J,ten,nine,A,K,nine,nine,K,P4,P3,
A,P1,P2,K,Q,Q,scat,A,J,ten,
nine,P4,K,A,P3,Q,J,J,nine,ten,
P1,P2,A,K,P4,Q,J,J,A,ten,
P3,nine,scat,K,A,Q,J,P4,ten,P1,
P2,A,K,ten,J,P3,Q,A,ten,nine,
P4,K,nine,J,A,ten,Q
},
reel 4:
{
wild,P1,P2,scat,P3,P4,A,K,K,Q,
J,ten,A,ten,nine,nine,P2,K,A,P1,
Q,P3,Q,J,A,ten,K,ten,P4,nine,
P2,A,Q,J,wild,P1,A,scat,P3,K,
J,ten,J,A,P2,Q,ten,nine,K,A,
J,P4,J,P1,ten,P3,A,K,P2,Q,
J,ten,A,ten,nine,K,nine,nine,A
},
reel 5:
{
wild,P1,P2,scat,P3,P4,A,K,Q,A,
J,A,ten,P3,P1,P2,wild,P4,A,A,
K,Q,nine,P3,A,K,P1,P2,P4,A,
Q,J,wild,P3,A,K,J,P4,P1,P2,
A,ten,Q,P3,A,K,nine,P4
}
}
Score Selected Hits = 11,998,935, RTP = 8.6530224043, Rate = 111
Combination Line 42 Col 15

In more detail the report shows exactly the same statistics as we calculated in Excel.

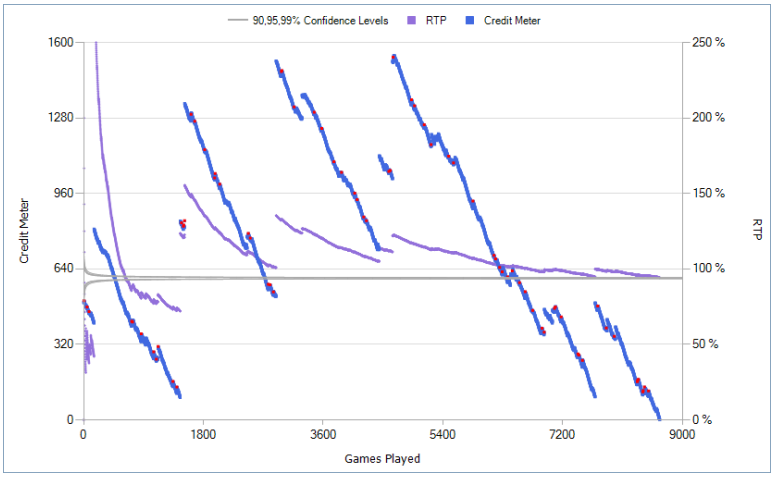


We can also easily examine many other game statistics.

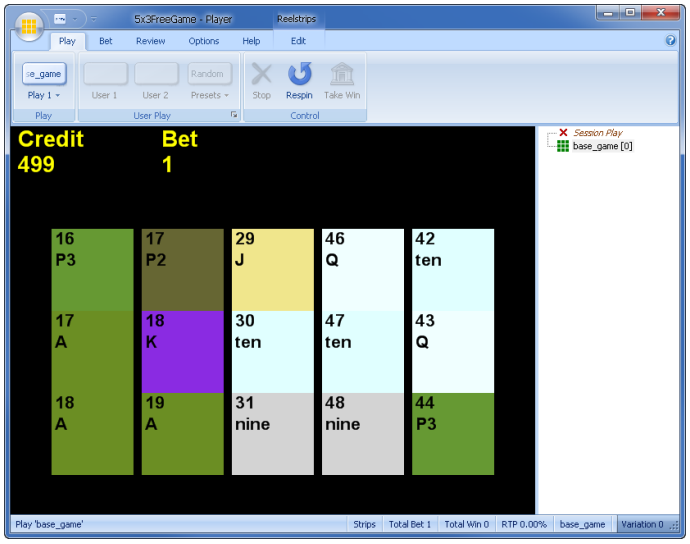
A random (Monte Carlo) simulation of the game, as if it were being played, shows the coinciding wins and how they are split between base and feature games (light/dark blue for base/feature respectively). These charts are from simulating the game for 100M plays over 12 seconds of simulation. This could also be calculated by a custom simulation in C++ or C# (for example), but not Excel.



We can also model play over a number of gaming sessions, and look at each session in detail. This particular session is very close to average, being only 0.03 standard deviations above the sample mean. The blue line indicates the credit meter, and the red dots are the credit meter when free game wins occurr.



A playable game is automatically generated from the mathematics enabling the game designer to get an immediate feel for how the game performs. Artwork can easily be added if required to further improve understanding of the game.



CHAPTER 9 SLOT DESIGNER

Slot Designer uses a text-based language to specify games, with the 3rd generation of this language (SD3) having been in use since 2008, and the 4th generation (SD4) introduced in late 2022.

While the product continued to improve, the SD4 project was started in 2013 to redesign Slot Designer to better support the vastly more complex games beginning to emerge.

This complexity is seen in Metamorphic games, which evolve during their play. A very simple example is a pay table multiplier that increments during the free games whenever a triggering combination appears on the screen. More complex examples would be those games commonly known[6] in the industry as Hold and Spin, Sticky Wild, Megaways, Shifting Reels, and Cascading Reels.

SD3 is a declarative language[7], that is, you state the problem, and the tool works out how to solve it.

The advantage of declarative programming is that you don't have to worry about the detailed problem of how to calculate the RTP of pay rules, wilds, scatters, etc. You don't even need to understand slot mathematics to use it. You state the rules of the game and Slot Designer can tell you the RTP via analysis, full cycle simulation, random simulation, or session simulation, and you can have a high degree of confidence that the game is correct. Or you can just play it.

And because the game model doesn't specify how to solve the game, huge advances in computational techniques have been possible.

SD4 is a new language that combines the best of the declarative language paradigm of SD3 with the imperative and object-oriented programming paradigms so popular in modern programming languages. Underlying and intricately linked to the language are a set of new mathematical algorithms that enable these problems to be efficiently solved. SD4 aims to simplify the design of *extremely* complex games.

Looking at games implemented in SD4 we can see the declarative simplicity of SD3, combined where necessary with the power and flexibility of features borrowed from C#, C++, and other languages.

By simplifying and speeding up the mathematical game design we ***reduce cost and time to market***, and by bringing game play right back to the early stages of game development we also ***improve game quality/monetisation***.

9.1 SD4 VERSION OF THE EXCEL GAME

The game we analyzed with Excel has screenshots showing SD3 code; here we can see the same game in SD4. For simple games it looks much the same.

```
symbol wild,scat,P1,P2,P3,P4,A,K,Q,J,ten,nine;
base g01;
meter<bool> is_feature = false;           // we start playing in the base game
class g01 : Game
{
   public construct()
   {
      window 5,3;
      payline line01 { 1,1,1,1,1 };
      if (is_feature)
         _bet = 0;
      scatter scat;
      substitute wild = all except scat;
      pay 1000,750,500    on left 5,4,3    P1;
```

6 Trademarks of various companies, including Aristocrat Technologies, Big Time Gaming, NetEnt, SG Gaming.

7 https://en.wikipedia.org/wiki/Comparison_of_programming_paradigms

```
        pay 750,500,100     on left 5,4,3   P2;
        pay 750,100,15      on left 5,4,3   P3;
        pay 500,40,15       on left 5,4,3   P4;
        pay 400,50,15       on left 5,4,3   A;
        pay 400,50,10       on left 5,4,3   K;
        pay 200,25,10       on left 5,4,3   Q;
        pay 200,25,10       on left 5,4,3   J;
        pay 150,25,10,5     on left 5,4,3,2 ten;
        pay 150,25,10,5     on left 5,4,3,2 nine;
        pay 100,20,10       on any  5,4,3   scat;
        pay 25*g01,15*g01,10*g01 on any 5,4,3 scat;
        reelstrips strips;
    }
    // Executed after game is played, and before next game.
    protected override void transition(TransitionContext& context) const
    {
        is_feature = true;
    }
}
const Reelstrips strips =
{
    {
        wild,P2,scat,P3,P4,A,K,Q,A,
        J,J,ten,nine,P1,P2,P3,A,A,K,
        Q,P4,J,ten,nine,A,scat,P1,P2,P3,
        K,K,Q,A,J,nine,P4,ten,K,nine,
        P1,P2,P3,A,K,Q,J,K,nine,nine,
        ten,A,scat,P1,P2,P3,P4,K,Q,A,
        Q,J,nine,ten,nine,nine,P1,P2,P3,A,
        K,P4,Q,Q,J,nine,ten
    },
    {
        wild,P2,scat,P3,P4,A,A,K,Q,
        J,ten,nine,A,K,nine,P1,P2,K,A,
        J,P3,K,Q,scat,ten,A,Q,P4,J,
        P1,P2,A,K,J,ten,nine,nine,A,P3,
        Q,J,K,J,P1,P2,A,scat,ten,ten,
        nine,P4,A,K,Q,J,P3,nine,P1,P2,
        A,K,ten,ten,J,Q,nine,A,nine,nine
    },
    {
        wild,P1,P2,scat,P3,P4,A,A,K,Q,
        J,ten,nine,A,K,nine,nine,K,P4,P3,
        A,P1,P2,K,Q,Q,scat,A,J,ten,
        nine,P4,K,A,P3,Q,J,J,nine,ten,
        P1,P2,A,K,P4,Q,J,J,A,ten,
        P3,nine,scat,K,A,Q,J,P4,ten,P1,
        P2,A,K,ten,J,P3,Q,A,ten,nine,
        P4,K,nine,J,A,ten,Q
    },
    {
        wild,P1,P2,scat,P3,P4,A,K,K,Q,
        J,ten,A,ten,nine,nine,P2,K,A,P1,
        Q,P3,Q,J,A,ten,K,ten,P4,nine,
        P2,A,Q,J,wild,P1,A,scat,P3,K,
        J,ten,J,A,P2,Q,ten,nine,K,A,
        J,P4,J,P1,ten,P3,A,K,P2,Q,
        J,ten,A,ten,nine,K,nine,nine,A
    },
    {
        wild,P1,P2,scat,P3,P4,A,K,Q,A,
        J,A,ten,P3,P1,P2,wild,P4,A,A,
        K,Q,nine,P3,A,K,P1,P2,P4,A,
        Q,J,wild,P3,A,K,J,P4,P1,P2,
        A,ten,Q,P3,A,K,nine,P4
    }
};
```

9.2 CHOOSE 2 OF 4 FEATURE

In Chapter 5 we looked at a game where 2 credit prizes are chosen from a set of 4. In SD3 we can model this with a built-in pay rule.

```
base { g01 }
game g01
{
   pay 1,2,3,4 on choose hidden 2;
}
```

In SD4, we can model the game at a more fundamental level. It looks much like a normal modern programming language, but it's not - the game can be analytically solved, randomly simulated, played, etc.

```
base g01;

meter<int[]> Chosen;            // the indicies in the prize pool of the prizes chosen so far
class g01 : Game
{
   const int _num_choices = 2;                  // pick 2 prizes
   const Prize[] _pool = { 1,2,3,4 };           // The prize pool
   protected override void play(PlayContext& context, Outcome& outcome) const
   {
      double[_pool.count()] weights;            // Build an array of weights for the RNG
      for (const int i in 0..weights.count())   // Initialse to 1
         weights[i] = 1;
      for (const int choice in Chosen)          // Don't reselect prior choices
         weights[choice] = 0;
      if (Chosen.empty())                       // On the first play set the bet.
         context.bet = 1;
      // Get a random index into the pool

      const int choice = context.rng.choose_hidden(0.._pool.count(), weights);
      Chosen.add(choice);                       // Remember the index for the next play
      outcome.wins.add(Win(_pool[choice]));     // Award the prize
      if (Chosen.count() < _num_choices)        // If there's any more choices then retrigger
      {
         outcome.wins.add(Win(g01));
      }
   }
}
```

While this previous game is more complex in SD4 than SD3, it allows enormous flexibility in designing features, and the complexity can be hidden away in a library. This is the same game implemented with the built-in SD4 library, and in a customised library it could be even more concise.

```
import std.GameChooseHidden;                    // import the standard library
base pool2of4;
class pool2of4 : std::GameChooseHidden
{
   public construct() :
      base(2,0,
      {
         std::Choice(1), std::Choice(2), std::Choice(3), std::Choice(4)
      })
   {
   }
}
```

9.3 SNAKES AND LADDERS

This Snakes and Ladders game is not possible in SD3 and is quite complex in Excel.

```
base sample::g01;            // the base game

namespace sample
{
   // Define two global meters

   meter<int> throws = 5;    // throws of the dice remaining
   meter<int> position = 0;  // position on the board

   class Square              // The Square class models a single square on the 10x10 grid.
   {
      const int _credit_win;
      public const int Advance;

      public construct()
      {
      }
      // adjust the position on the board, i.e. snake or ladder
      public construct(int advance, int credit_win)
      {
         Advance = advance;
         _credit_win = credit_win;
      }
      public void play(int& position_meter, Outcome& outcome)
      {
         position_meter += Advance;
         if (_credit_win != 0)
         {
            outcome.wins.add(Win(_credit_win));
         }
      }
   }
   class g01 : Game
   {
      // The board is a 10x10 grid, and defined upside down as position 0 is top left while in
      // practice its bottom left.
      const Square[] _board =
      {
         Square(),       Square(8,0),    Square(0,1),    Square(),       Square(),
         Square(),       Square(),       Square(),       Square(),       Square(0,1),
         Square(),       Square(),       Square(),       Square(),       Square(),
         Square(),       Square(0,1),    Square(),       Square(),       Square(),
         Square(0,5),    Square(),       Square(),       Square(0,5),    Square(),
         Square(55,0),   Square(),       Square(),       Square(),       Square(),
         Square(),       Square(),       Square(0,10),   Square(),       Square(8,0),
         Square(),       Square(),       Square(),       Square(),       Square(),
         Square(),       Square(),       Square(),       Square(),       Square(),
         Square(),       Square(0,10),   Square(),       Square(),       Square(),
         Square(),       Square(),       Square(),       Square(0,20),   Square(-19,0),
         Square(),       Square(),       Square(),       Square(),       Square(),
         Square(),       Square(),       Square(),       Square(),       Square(),
         Square(),       Square(),       Square(),       Square(),       Square(),
         Square(0,100),  Square(),       Square(),       Square(),       Square(),
         Square(),       Square(),       Square(),       Square(),       Square(),
         Square(),       Square(),       Square(),       Square(),       Square(12,100),
         Square(0,100),  Square(),       Square(),       Square(),       Square(),
         Square(0,500),  Square(),       Square(),       Square(0,500),  Square(),
         Square(),       Square(0,700),  Square(0,700),  Square(0,700),  Square(0,10000)
      };
      protected override void play(PlayContext& context, Outcome& outcome) const
      {
         if (position == 0)
            context.bet = 100;                              // bet on the first square
         throws--;
         position += context.rng.next(6) + 1;               // throw the 'dice'
         position = std::min(position, _board.count()-1);   // don't go off end of board
         if (throws > 0 && position < _board.count()-1)     // don't go off the end of the board
            outcome.wins.add(Win(this'first));              // retrigger to play again
         bool advances = _board[position].Advance != 0;
```

```
            _board[position].play(&position, &outcome);
            // evaluate prize at the end of the snake or ladder

            if (advances)
               _board[position].play(&position, &outcome);
         }
      }
   }
```

9.4 METAMORPHIC HOLD AND SPIN

Hold and Spin games are relatively simple examples of metamorphic games, this one comprising 433 individual games. Again, it can be analytically solved, simulated, etc.

```
symbol scat, P1, P2, P3, P4, A, K, Q, J, ten, nine;
base base_game;

// The reelstrips for the game are stored in a meter, so that we can modify them from spin to spin
// of the free games. The meter is reset to the its default value (as given here) when the play
// the play button is pressed (just prior to to the base game).

meter<Reelstrips> reelstrips_meter =
{
   { P1,P2,P3,P4,A,A,K,Q,J,scat,P1,P2,P3,P4,A,A,K,Q,J,Q },
   { P1,P2,P3,P4,A,A,K,Q,J,scat,P1,P2,P3,P4,A,A,K,Q,J,Q,ten,J,nine,A,Q,nine,nine,K },
   { P1,P2,P3,P4,A,A,K,Q,J,scat,P1,P2,P3,P4,A,A,K,Q,J,Q,A,K,ten,nine,nine,J,A },
   { P1,P2,P3,P4,A,A,K,Q,J,scat,P1,P2,P3,P4,A,A,K,Q,J,nine,nine,A,Q,J,ten,ten },
   { P1,P2,P3,P4,A,A,K,Q,J,scat,P1,P2,P3,P4,A,A,K,Q,J,Q }
};

// The custom hold prize modifes the reelstrips_meter when a hit occurrs on its pay rule.

class Hold : Prize
{
   const int _reel;
   const Symbol[] _pattern;

   public construct(int reel, Symbol[] pattern)
   {
      _reel = reel;
      _pattern = pattern;
   }

   public override void commit(PrizeContext& context) const
   {
      reelstrips_meter.reels[_reel].hold = true;
      reelstrips_meter.reels[_reel].symbols = _pattern;
   }
}

class common_game : Game
{
   protected construct(GameContext& context)
   {
      window 5,3;
      scatter scat;

      payline line01 { 1,1,1,1,1 };

      pay 300,20,2     on any  5,4,3   scat;
      pay 1000,500,75  on left 5,4,3   P1;
      pay 600,40,25    on left 5,4,3   P2;
      pay 500,40,15    on left 5,4,3   P3;
      pay 500,40,15    on left 5,4,3   P4;
      pay 400,25,15    on left 5,4,3   A;
      pay 400,25,10    on left 5,4,3   K;
      pay 200,25,10    on left 5,4,3   Q;
      pay 200,25,10    on left 5,4,3   J;
      pay 150,25,5,3   on left 5,4,3,2 ten;
      pay 150,25,5,3   on left 5,4,3,2 nine;

      _pays += createHoldPayRules(&reelstrips_meter);

      reelstrips reelstrips_meter;
   }

   // Create pay rules that for all the possible patterns of symbols that occur when the hold
   // trigger symbol (scat) hits, and custom prizes that copy those combinations into the
   // reelstrips.
```

```
    protected Pay[] createHoldPayRules(const Reelstrips& strips) const
    {
        Pay[] pay_rules;
        for (int reel in 0..strips.reels.count())     // for each of the reels
        {
            // Only create pay rules for reels that are not already held
            if (!strips.reels[reel].hold)
            {
                // A dictionary is like an array (of bool's in this case), but its indexed by an array
                // of symbols rather than an int. Here's its used to record each unique array of 3
                // symbols that occurr on the play window with a scat.
                dictionary<Symbol[], bool> patterns;
                int length = strips.reels[reel].symbols.count();

                for (int position in 0..length)
                {
                    Symbol[] pattern = getSymbols(strips.reels[reel].symbols[%position..position+3]);

                    if (pattern.contains(scat))
                    {
                        if (!patterns.containsKey(pattern))    // don't create duplicate pay rules
                        {
                            patterns[pattern] = true;
                            pay_rules += pay Hold(reel, pattern) on pattern @ {reel:0, reel:1, reel:2};
                        }
                    }
                }
            }
        }
        return pay_rules;
    }
    // Reels are an array of WeightedSymbol, not Symbol. Each stop position can have multiple
    // symbols. Extract the symbol from each position. We don't consider multiple symbols as is not
    // common and make this sample harder to understand.
    Symbol[] getSymbols(WeightedSymbol[] weighted) const
    {
        Symbol[] pattern;
        for (WeightedSymbol sym in weighted)
        {
            if (sym.symbols.count() != 1)
                error "Compound symbols not allowed";
            pattern.add(sym.symbols[0]);
        }
        return pattern;
    }
}
// In the base game if we get left 3,4, or 5 scatters we trigger 3 free games.

class base_game : common_game
{
    public construct(GameContext& context) :
        base(&context)
    {
        pay 3 * respin on left 5,4,3 scat @ {0:1,1:1,2:1,3:1,4:1};
    }
}
class respin : common_game
{
    public construct(GameContext& context) :
        base(&context)
    {
        _bet = 0;
    }
}
```

CHAPTER 10 ACKNOWLEDGEMENTS

Sections of this book were developed with the aid of the following people

Mark Sinosich of Imagine Numbers.

http://www.imaginenumbers.com/

Contributed Chapter 3 “243 Ways Games”, page 17.