NOVA 6

Johnathan Napier

Preliminary Discussion on Combination versus Permutation

In the next lines of code, I'll be sort of "brute-forcing" the probabilities. Yes, we could go through them the rigorous "mathy" way, but the calculation portion will be demonstratively shorter. However, I will go into the theory if you still wanted to do some calculations on your own. Whenever you employ a concept of "better" or "worse," removing/adding die, or even the utilization of "sum;" the mathematics gets stupidly-obnoxious! Regardless, here is an explanation of combination/permutation. We'll use horse-races and poker as prime examples.

Factorial

You'll often see an exclamation point written in these formulas. Mathematics latched onto this symbol (factorial) to mean a sequential product:

$3! = 1 \cdot 2 \cdot 3 \qquad 7! = 1 \cdot 2 \cdot 3 \cdot 4 \cdot 5 \cdot 6 \cdot 7$

We use this when the number or things we want to pick is the same as the number of options. How many ways are there to shuffle 52 cards in a poker deck? There are 52! ways. This is because we have 52 choices for the first card. Once that card is locked in, we have 51 choices for the next, and so on. The same goes for horse races: 12 horses, they each have to come in a place: 1st - 12th. So, 12 choices for 1st, then 11 choices for 2nd, and so on.

Permuation: Order matters!

Things get a little different if we don't use up all the possibilities. Let's say I want to compute the different ways of assembling a trifecta in horse races. We have 12 for the first choice, 11 for the second, 10 for the third, ... and then we stop. Notation wise, you will see it written a number of ways: $P_k^n = P(n,k) = {}_nP_k = \frac{n!}{(n-k)!}$. It is read "n permute k." So, for the trifecta:

$P_3^{12} = P(12,3) = {}_{12}P_3 = \frac{12!}{(12-3)!} = \frac{12!}{9!} = \frac{12 \cdot 11 \cdot 10 \cdot \cancel{9 \cdot 8 \cdot \ldots \cdot 2 \cdot 1}}{\cancel{9 \cdot 8 \cdot \ldots \cdot 2 \cdot 1}} = 12 \cdot 11 \cdot 10$

Combination: Order doesn't matter.

Now let's consider cases where don't care about order. Poker is a perfect example. No one cares if the Jack of Diamonds is first in your hand because you can rearrange them and they have the same value. So, you have to account for this through division. You select the cards first from the total population. Then once you have them, you account for all the orders in which they can come. Notation-wise, you'll see it written: $\binom{n}{k} = C_k^n = {}_nC_k = \frac{n!}{k!(n-k)!}$. It is read "n choose k." This new term of k! in the denominator accounts for all the ways to arrange such a collection. So, let's do some examples:

Number of 5-card hands: $\binom{52}{5} = C_5^{52} = {}_{52}C_5 = \frac{52!}{5!(52-5)!} = \frac{52 \cdot 51 \cdot 50 \cdot 49 \cdot 48}{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1} = \frac{311875200}{120} = 2598960$

Number of full houses: ${}_{13}C_1 \cdot {}_4C_3 \cdot {}_{12}C_1 \cdot {}_4C_2 = 13 \cdot 4 \cdot 12 \cdot 6 = 3744$

Number of flushes: ${}_4C_1 \cdot {}_{13}C_5 = 4 \cdot 1287 = 5148$

Number of ways to "box" the trifecta: ${}_{12}C_3 = 220$

Assembling Roll Possibilities

A trick that I am employing here is to roll 7 dice with values from 0 to 6. Wait! What?!? We can treat a die with value of zero as a die that you didn't roll. It's a space filler. So, we need arrays of length 7 that can take on 7 different values, making 7^7 possibilities. Obviously, some of these would be infeasible and/or replicate. (3,4,0,0,0,0,0) would mean you rolled 2 dice which is not possible. Also, (3,4,1,0,0,0,0) would be replicate to (0,0,0,0,3,4,1), since it doesn't matter what order the filler die come in. However, we do want keep things like (1,1,1,2,2,2,4) and (1,2,1,2,1,2,4) for probability purposes.

```
# Creates an array of zeros: 7 dice valued at 0 and a count of 0
possible.dice<-data.frame(t(rep(0,7)),number.dice=0)
for(i in 1:7){
  for(j in 1:6){
    temp<-possible.dice # create a copy of the data.frame
    temp[,i]<-j # replace the value of a die
    temp$number.dice<-rowSums(temp[,1:7] != 0) # recount the number on "non-zero" die
    possible.dice<-rbind(possible.dice,temp) # append to old list of possibilities
    possible.dice<-possible.dice[!duplicated(possible.dice),] # remove repeats
  }
}
# this leaves 7^7 = 823,543 possibilities

possible.dice<-subset(possible.dice,number.dice>=3) # reduce down to 3-7 dice: 822,744 possible
for(i in 1:7){# push the zeros to the end
  index<-which(possible.dice$X1==0)
  possible.dice[index,1:6]<-possible.dice[index,2:7]
  possible.dice[index,7]<-0

  index<-which(possible.dice$X2==0)
  possible.dice[index,2:6]<-possible.dice[index,3:7]
  possible.dice[index,7]<-0

  index<-which(possible.dice$X3==0)
  possible.dice[index,3:6]<-possible.dice[index,4:7]
  possible.dice[index,7]<-0

  index<-which(possible.dice$X4==0)
  possible.dice[index,4:6]<-possible.dice[index,5:7]
  possible.dice[index,7]<-0

  index<-which(possible.dice$X5==0)
  possible.dice[index,5:6]<-possible.dice[index,6:7]
  possible.dice[index,7]<-0

  index<-which(possible.dice$X6==0)
  possible.dice[index,6]<-possible.dice[index,7]
  possible.dice[index,7]<-0
}
possible.dice<-possible.dice[!duplicated(possible.dice),] # remove repeats: 335,880 possible
save.image()
```

Best and Worst

So, we haven't even gotten to up, down, or even yet. The great thing is that we can do them all at once. We'll choose the "best" and "worst" set of 3 dice for the given possibilities as one dataset.

```
library(dplyr,quietly = TRUE)
rownames(possible.dice)<-NULL
possible.dice$high.A1<-NA # filler for action dice 1 (high)
possible.dice$high.A2<-NA # filler for action dice 2 (high)
possible.dice$high.A3<-NA # filler for action dice 3 (high)
possible.dice$low.A1<-NA # filler for action dice 1 (low)
possible.dice$low.A2<-NA # filler for action dice 2 (low)
possible.dice$low.A3<-NA # filler for action dice 3 (low)
possible.dice$high.result<-0 # filler for total (high)
possible.dice$low.result<-99 # filler for total (low)
possible.dice$high.sdt<--3 # filler for single/double/triple status (high)
possible.dice$low.sdt<-3 # filler for single/double/triple status (low)
possible.dice$high.success<-0 # filler for success/failure status (high)
possible.dice$low.success<-1 # filler for success/failure status (low)
for(i in 1:5){
  for(j in (i+1):6){
    for(k in (j+1):7){
      these.dice<-possible.dice[,c(i,j,k)] # pick 3 dice based on column-combination
      total<-rowSums(these.dice) # take the sum of dice
      success<-total>=11 # success or failure with these dice
      # next we get the single/double/triple
      sdt<-(these.dice[,1]==these.dice[,2])*2+(these.dice[,1]!=these.dice[,2])*1
      sdt<-pmax(sdt,(these.dice[,1]==these.dice[,3])*2)
      sdt<-pmax(sdt,(these.dice[,2]==these.dice[,3])*2)
      sdt<-pmax(sdt,((these.dice[,1]==these.dice[,2])&(these.dice[,1]==these.dice[,3]))*3)
      sdt<-sdt*(success==0)*-1+sdt*(success==1)*1
      infeasible<-(these.dice==0) %>% rowSums() # are there three dice or not?
      update.high<-which(infeasible==0 & # which rows will we update? (high)
                         success>=possible.dice$high.success &
                         sdt>=possible.dice$high.sdt)
      update.low<-which(infeasible==0 & # which rows will we update? (low)
                        total<=possible.dice$low.result)
      possible.dice[update.high,c("high.A1","high.A2","high.A3")]<-these.dice[update.high,]
      possible.dice[update.low,c("low.A1","low.A2","low.A3")]<-these.dice[update.low,]
      possible.dice[update.high,"high.result"]<-total[update.high]
      possible.dice[update.low,"low.result"]<-total[update.low]
      possible.dice[update.high,"high.success"]<-success[update.high]
      possible.dice[update.low,"low.success"]<-success[update.low]
      possible.dice[update.high,"high.sdt"]<-sdt[update.high]
      possible.dice[update.low,"low.sdt"]<-sdt[update.low]
    }
  }
}
```

Rerun of Best/Worst

While this is not too technically important, the previous code gave preference to movement between statuses.

High: (Fail-3) → (Fail-2) → (Fail-1) → (Success-1) → (Success-2) → (Success-3)

However, this approach overwrites the total for high rolls. Now we will run again, keeping the same status (when up only), but shifting in dice total. We will also make sure that the lowest 3 dice are selected when down, given that for example $6+5+1=4+4+4$ have the same total value.

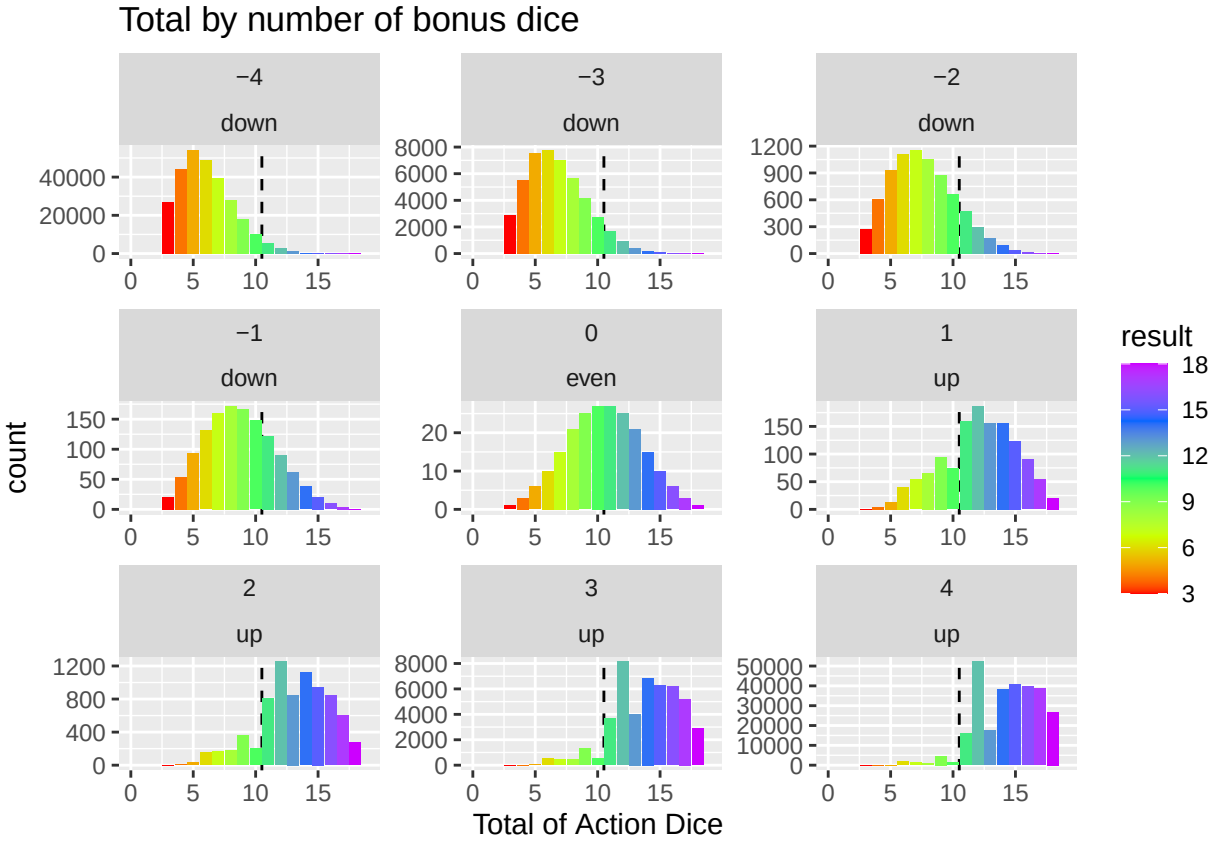
```
library(dplyr,quietly=TRUE)
for(i in 1:5){
  for(j in (i+1):6){
    for(k in (j+1):7){
      these.dice<-possible.dice[,c(i,j,k)] # pick 3 dice based on column-combination
      total<-rowSums(these.dice) # take the sum of dice
      success<-total>=11 # success or failure with these dice
      # next we get the single/double/triple
      sdt<-(these.dice[,1]==these.dice[,2])*2+(these.dice[,1]!=these.dice[,2])*1
      sdt<-pmax(sdt,(these.dice[,1]==these.dice[,3])*2)
      sdt<-pmax(sdt,(these.dice[,2]==these.dice[,3])*2)
      sdt<-pmax(sdt,((these.dice[,1]==these.dice[,2])&(these.dice[,1]==these.dice[,3]))*3)
      sdt<-sdt*(success==0)*-1+sdt*(success==1)*1
      infeasible<-(these.dice==0) %>% rowSums() # are there three dice or not?
      update.high<-which(infeasible==0 & # which rows will we update? (high)
                         success>=possible.dice$high.success &
                         sdt>=possible.dice$high.sdt &
                         total>=possible.dice$high.result) # modified here
      top.die<-these.dice %>% apply(1,max) # modified here
      low.die<-these.dice %>% apply(1,min) # modified here
      top.die.prior<-possible.dice[,c("low.A1","low.A2","low.A3")] %>% apply(1,max) # modified here
      low.die.prior<-possible.dice[,c("low.A1","low.A2","low.A3")] %>% apply(1,min) # modified here
      update.low<-which(infeasible==0 & # which rows will we update? (low)
                        total<=possible.dice$low.result &
                        top.die<=top.die.prior & # modified here
                        low.die<=low.die.prior) # modified here
      possible.dice[update.high,c("high.A1","high.A2","high.A3")]<-these.dice[update.high,]
      possible.dice[update.low,c("low.A1","low.A2","low.A3")]<-these.dice[update.low,]
      possible.dice[update.high,"high.result"]<-total[update.high]
      possible.dice[update.low,"low.result"]<-total[update.low]
      possible.dice[update.high,"high.success"]<-success[update.high]
      possible.dice[update.low,"low.success"]<-success[update.low]
      possible.dice[update.high,"high.sdt"]<-sdt[update.high]
      possible.dice[update.low,"low.sdt"]<-sdt[update.low]
    }
  }
}
```

Visualization

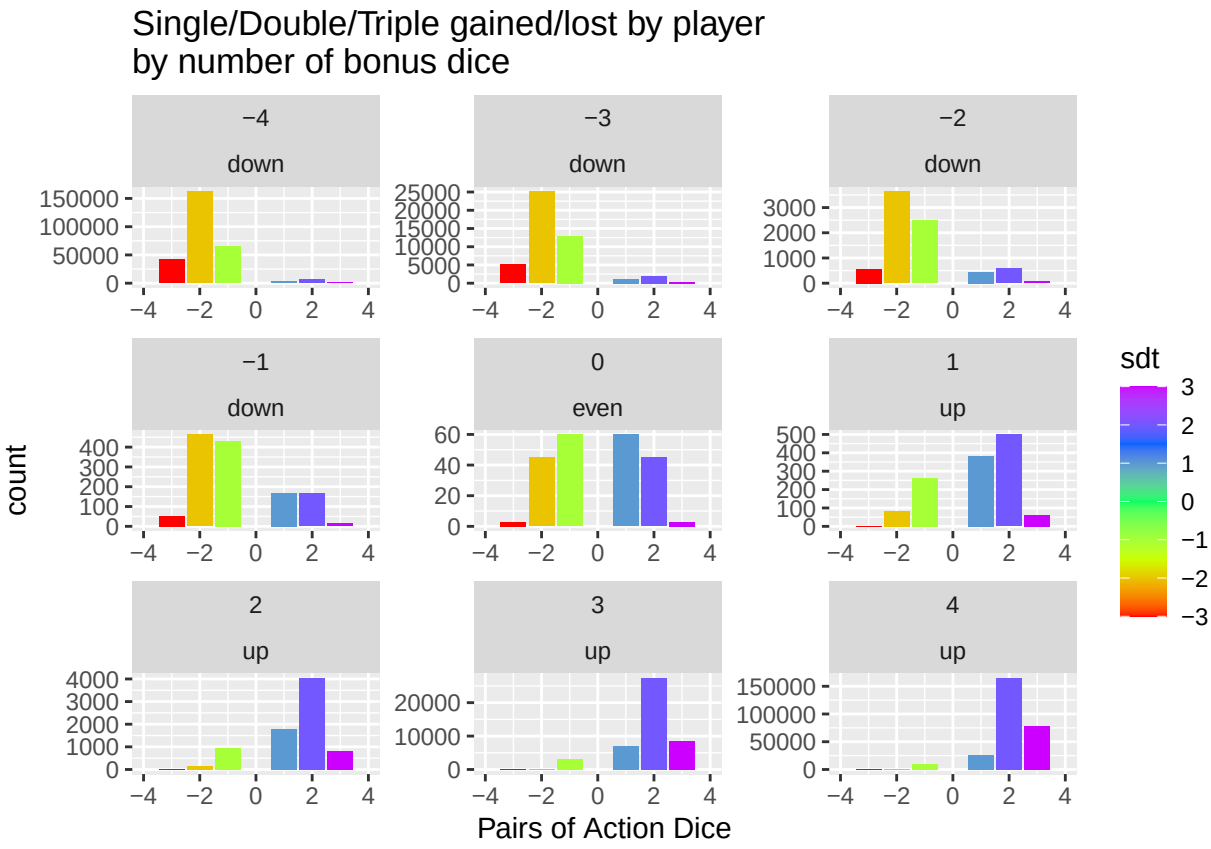
Now lets look at the results and get a visceral feel for what can happen. We'll consider single/double/triple as 1/2/3 if the player wins or -1/-2/-3 if the GM wins. Bonus dice is simply number of dice - 3 when up (or even), as well as 3 - number of dice when down (or even). Thus, bonus dice ranges from -4 to 4.

```
library(ggplot2)
up<-possible.dice[,1:8]
up$A1<-possible.dice$high.A1
up$A2<-possible.dice$high.A2
up$A3<-possible.dice$high.A3
```

```
up$result<-possible.dice$high.result
up$success<-possible.dice$high.success
up$sdt<-possible.dice$high.sdt
up$bonus.dice<-up$number.dice-3
up<-subset(up,number.dice>=4)
up$mode<-"up"

down<-possible.dice[,1:8]
down$A1<-possible.dice$low.A1
down$A2<-possible.dice$low.A2
down$A3<-possible.dice$low.A3
down$result<-possible.dice$low.result
down$success<-possible.dice$low.success
down$sdt<-possible.dice$low.sdt
down$bonus.dice<-3-down$number.dice
even<-subset(down,number.dice==3)
down<-subset(down,number.dice>=4)
even$mode<-"even"
down$mode<-"down"

aggregate<-rbind(up,even,down)

ggplot(data=aggregate,aes(x=result,fill=result,group=result))+
  geom_vline(xintercept = 10.5,lty=2)+facet_wrap(vars(bonus.dice,mode),scales="free")+
  geom_bar()+scale_fill_gradientn(colors = rainbow(5))+
  xlab("Total of Action Dice")+xlim(0,19)+ggtitle("Total by number of bonus dice")
```



```
ggplot(data=aggregate,aes(x=sdt,fill=sdt,group=sdt))+
  facet_wrap(vars(bonus.dice,mode),scales="free")+
  geom_bar()+scale_fill_gradientn(colors = rainbow(5))+
  xlab("Pairs of Action Dice")+xlim(-4,4)+
  ggtitle("Single/Double/Triple gained/lost by player\nby number of bonus dice")
```



Probability

Now that all the hard work is done, we can final get to the numbers. We'll do a two-way table of frequencies/probabilities. However, it will be important to keep things separated by bonus dice as the numbers will otherwise be misleading. We'll have to divide by the sum of each column in order to get the proper probabilities of each situation. The numbers of interest will be success/failure, raw total of action dice, single/double/triple, and single/double/triple when it goes to the player/GM.

```
options(scipen=999)
library(dplyr)
library(kableExtra)
# Frequency/Probability of Success by Bonus Dice
tab1<-table(aggregate$success,aggregate$bonus.dice)
kable(addmargins(tab1,1))
```

	-4	-3	-2	-1	0	1	2	3	4
0	269650	43305	6688	947	108	349	1088	3351	10286
1	10286	3351	1088	349	108	947	6688	43305	269650
Sum	279936	46656	7776	1296	216	1296	7776	46656	279936

```
kable((addmargins(t(t(tab1)/colSums(tab1)),1)*100) %>% round(7))
```

	-4	-3	-2	-1	0	1	2	3	4
0	96.325589	92.817644	86.00823	73.07099	50	26.92901	13.99177	7.182356	3.674411
1	3.674411	7.182356	13.99177	26.92901	50	73.07099	86.00823	92.817644	96.325589
Sum	100.000000	100.000000	100.00000	100.00000	100	100.00000	100.00000	100.000000	100.000000

```
# Frequency/Probability of Total by Bonus Dice
tab2<-table(aggregate$result,aggregate$bonus.dice)
kable(addmargins(tab2,1))
```

	-4	-3	-2	-1	0	1	2	3	4
3	26811	2906	276	21	1	1	1	1	1
4	44121	5535	610	54	3	4	5	6	7
5	54096	7551	935	94	6	14	30	62	126
6	48798	7770	1111	131	10	41	156	547	1814
7	39277	7056	1155	160	15	54	165	458	1197
8	27979	5646	1055	172	21	66	175	428	1001
9	18159	4135	881	167	25	95	356	1312	4719
10	10409	2706	665	148	27	74	200	537	1421
11	5747	1677	470	122	27	160	805	3696	15911
12	2745	914	296	91	25	187	1251	8137	52151
13	1141	447	170	62	21	156	850	4020	17619
14	469	207	90	38	15	156	1120	6855	38031
15	148	78	41	21	10	123	946	6261	40692
16	28	21	15	10	6	90	840	6210	39690
17	7	6	5	4	3	54	600	5220	38745
18	1	1	1	1	1	21	276	2906	26811
Sum	279936	46656	7776	1296	216	1296	7776	46656	279936

```
kable((addmargins(t(t(tab2)/colSums(tab2)),1)*100) %>% round(7)) %>%
  kable_styling(font_size = 7,position = "left")
```

	-4	-3	-2	-1	0	1	2	3	4
3	9.5775463	6.2285665	3.5493827	1.6203704	0.462963	0.0771605	0.0128601	0.0021433	0.0003572
4	15.7611025	11.8634259	7.8446502	4.1666667	1.388889	0.3086420	0.0643004	0.0128601	0.0025006
5	19.3244170	16.1844136	12.0241770	7.2530864	2.777778	1.0802469	0.3858025	0.1328875	0.0450103
6	17.4318416	16.6538066	14.2875514	10.1080247	4.629630	3.1635802	2.0061728	1.1724108	0.6480053
7	14.0307070	15.1234568	14.8533951	12.3456790	6.944444	4.1666667	2.1219136	0.9816529	0.4275977
8	9.9947845	12.1013374	13.5673868	13.2716049	9.722222	5.0925926	2.2505144	0.9173525	0.3575817
9	6.4868398	8.8627401	11.3297325	12.8858025	11.574074	7.3302469	4.5781893	2.8120713	1.6857425
10	3.7183499	5.7998971	8.5519547	11.4197531	12.500000	5.7098765	2.5720165	1.1509774	0.5076160
11	2.0529693	3.5943930	6.0442387	9.4135802	12.500000	12.3456790	10.3523663	7.9218107	5.6837992
12	0.9805813	1.9590192	3.8065844	7.0216049	11.574074	14.4290123	16.0879630	17.4404150	18.6296153
13	0.4075932	0.9580761	2.1862140	4.7839506	9.722222	12.0370370	10.9310700	8.6162551	6.2939386
14	0.1675383	0.4436728	1.1574074	2.9320988	6.944444	12.0370370	14.4032922	14.6926440	13.5856053
15	0.0528692	0.1671811	0.5272634	1.6203704	4.629630	9.4907407	12.1656379	13.4194959	14.5361797
16	0.0100023	0.0450103	0.1929012	0.7716049	2.777778	6.9444444	10.8024691	13.3101852	14.1782407
17	0.0025006	0.0128601	0.0643004	0.3086420	1.388889	4.1666667	7.7160494	11.1882716	13.8406636
18	0.0003572	0.0021433	0.0128601	0.0771605	0.462963	1.6203704	3.5493827	6.2285665	9.5775463
Sum	100.0000000	100.0000000	100.0000000	100.0000000	100.000000	100.0000000	100.0000000	100.0000000	100.0000000

```
# Frequency/Probability of Single/Double/Triple by Bonus Dice
# This is irrespective of wins/losses
tab3<-table(abs(aggregate$sdt),aggregate$bonus.dice)
kable(addmargins(tab3,1))
```

	-4	-3	-2	-1	0	1	2	3	4
1	67620	14100	2920	600	120	648	2740	10230	35266
2	168910	27135	4225	630	90	582	4205	27765	166124
3	43406	5421	631	66	6	66	831	8661	78546
Sum	279936	46656	7776	1296	216	1296	7776	46656	279936

```
kable((addmargins(t(t(tab3)/colSums(tab3)),1)*100) %>% round(7)) %>%
  kable_styling(font_size = 7,position = "left")
```

	-4	-3	-2	-1	0	1	2	3	4
1	24.15552	30.22119	37.551440	46.296296	55.555556	50.000000	35.23663	21.92644	12.59788
2	60.33879	58.15972	54.333848	48.611111	41.666667	44.907407	54.07665	59.51003	59.34356
3	15.50569	11.61908	8.114712	5.092593	2.777778	5.092593	10.68673	18.56353	28.05856
Sum	100.00000	100.00000	100.000000	100.000000	100.000000	100.000000	100.00000	100.00000	100.00000

```
# Frequency/Probability of Single/Double/Triple Gain/Loss by Bonus Dice
# Here we separate by wins/losses
tab4<-table(aggregate$sdt,aggregate$bonus.dice)
kable(addmargins(tab4,1))
```

	-4	-3	-2	-1	0	1	2	3	4
-3	42367	5145	563	51	3	3	3	3	3
-2	162603	25200	3645	464	45	82	145	258	469
-1	64680	12960	2480	432	60	264	940	3090	9814
1	2940	1140	440	168	60	384	1800	7140	25452
2	6307	1935	580	166	45	500	4060	27507	165655
3	1039	276	68	15	3	63	828	8658	78543
Sum	279936	46656	7776	1296	216	1296	7776	46656	279936

```
kable((addmargins(t(t(tab4)/colSums(tab4)),1)*100) %>% round(7)) %>%
  kable_styling(font_size = 7,position = "left")
```

	-4	-3	-2	-1	0	1	2	3	4
-3	15.1345307	11.0275206	7.2402263	3.935185	1.388889	0.2314815	0.0385802	0.0064300	0.0010717
-2	58.0857767	54.0123457	46.8750000	35.802469	20.833333	6.3271605	1.8647119	0.5529835	0.1675383
-1	23.1052812	27.7777778	31.8930041	33.333333	27.777778	20.3703704	12.0884774	6.6229424	3.5058013
1	1.0502401	2.4434156	5.6584362	12.962963	27.777778	29.6296296	23.1481481	15.3034979	9.0920782
2	2.2530150	4.1473765	7.4588477	12.808642	20.833333	38.5802469	52.2119342	58.9570473	59.1760259
3	0.3711563	0.5915638	0.8744856	1.157407	1.388889	4.8611111	10.6481481	18.5570988	28.0574846
Sum	100.0000000	100.0000000	100.0000000	100.000000	100.000000	100.0000000	100.0000000	100.0000000	100.0000000