A Consideration of Probability and Runs Scored Major League Baseball Extra-Inning Games

Paul A. Hodgetts

02/03/2021

Abstract

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Nisl suscipit adipiscing bibendum est ultricies integer quis auctor elit. Facilisis leo vel fringilla est. Dignissim suspendisse in est ante in. Condimentum vitae sapien pellentesque habitant. Dui vivamus arcu felis bibendum ut tristique et egestas. Orci sagittis eu volutpat odio facilisis mauris.

Introduction

The COVID-19 pandemic led to many questions and concerns in the sporting world, namely whether leagues would commit to hosting a 2020 season, and if so how to do so safely? For the league leadership that made the decision to host a 2020 season, various protocols were required to ensure the health and safety of athletes and staff. For example, the National Hockey League (NHL) implemented a bubble system, with all teams within the Western Conference playing within Edmonton, Alberta, and all teams within the Eastern Conference within Toronto, Ontario (Gatto, 2020). In a similar move, the National Basketball Association (NBA) established a bubble in Orlando, Florida within which teams could play out the remainder of the season (Haislop, 2020). However, unlike the use of a bubbled league like the NHL and NBA, Major League Baseball (MLB) permitted teams to play games within their own stadiums, excluding the Toronto Blue Jays who were denied access to play within Canada by the Canadian federal government (McNamara, 2020; Wagner, 2020). In choosing this approach, MLB implemented policies such as no spitting, masks being required in the dugout and bullpen, no use of saunas, twice-a-day temperature and symptom checks, and visiting team members were to remain bubbled within the hotel (Wagner, 2020). Another such policy was to also introduce a new rule regarding extra-inning games, tied games that go beyond the regulation nine innings, in hopes of shortening the exposure experienced by players between teams (Allen, 2020). The rule was that if a game were to reach extra-innings, then each subsequent half-inning would begin with each team having a runner on second-base (Allen, 2020). The idea being that this would increase the probability of either team scoring in the half-inning, thereby shortening the amount of time athletes spent on the diamond. Yet, while this rule change was implemented as a protective measure toward the athlete, a rule change to a sport or game should also be done to ensure the fairness of the playing field. This paper looks to explore this rule change for the 2020 MLB season and whether it provides an advantage to the visitor team in extra-inning games by increasing the probability of scoring a run. As well, it is considered whether extra-inning games in general provide an advantage to the visitor team and how strategy might change in extra-inning games with and without the new rule. It additionally explores the analysis behind how this rule change would protect players by limiting on-field exposure between teams, and whether this was accomplished by shortening extra-inning games as compared to previous seasons.

Data

Access

This analysis uses game-log and play-by-play data for each MLB season from the 2000 MLB season to the 2020 MLB season. Game-logs and play-by-play files were obtained free of charge from and are copyrighted by Retrosheet (2021). Interested parties may contact Retrosheet at www.retrosheet.org.. The play-by-play files were accessed using the `parse_retrosheet_pbp()` function as written by GitHub user beanumber (Baumer, 2019), with the process described by Marchi et al. (2019c) in 'Appendix A' of 'Analyzing Baseball Data with R'. Other data files used include the fields dataset maintained by GitHub user maxtoki (Max, 2013). This dataset provides the Retrosheet data event headers and can be accessed from the baseball_R GitHub repository.

Missing Values

Regarding game-log data, four missing values were found in regulation length games and two missing values were found in extra-inning games due to tie-games. These values were removed from their respective datasets leaving N = 45,283 observations in regulation length games and N = 4,197 in extra-inning games across all seasons. Play-by-play files were also examined for missing values; however, all missing values belonged to event-type variables (e.g. the play on runner on second). These values were not removed as doing so would create issues within the data regarding analysis.

Exploratory Analysis

An exploratory analysis revealed that over all seasons the home team won more games, for both extra-inning games and regulation games (Figure 1). Analysing regulation length games by individual season also showed that across all seasons the home team won more games than the visiting team in regulation length games (Figure 2 and Table 1). Additionally, analysing each extra-inning game by individual season revealed that generally the same pattern of the home team winning more games than the visitor, with the home team winning more games in 15 of 21 seasons (Figure 3 and Table 2). However, in the 2000 and 2012 seasons both the home team and visitor won an equal number of extra-inning games with N = 101 and N = 96 games respectively (Figure 3 and Table 2). Meanwhile, the visitor won more extra-inning games in 2001 (Visitor, N = 101; Home, N = 94), 2014 (Visitor, N = 120; Home, N = 112), 2019 (Visitor, N = 109; Home, N = 99), and 2020 (Visitor, N = 36; Home, N = 32) (Figure 3 and Table 2).

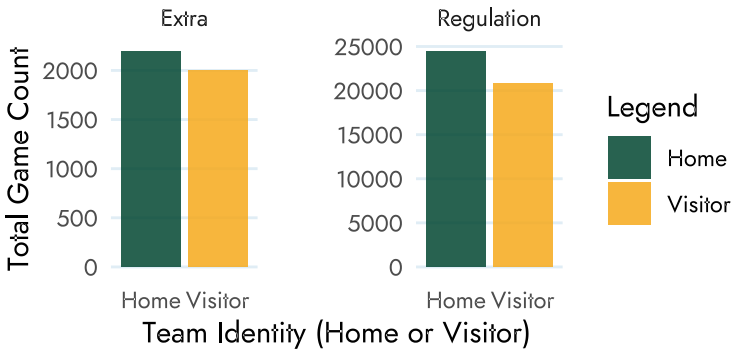

Figure 1: Home vs. Visitor Wins for Extra-Inning Games and Regulation Length Games for MLB Seasons 2000-2020

Figure 2: Home vs. Visitor Wins for Regulation Length Games for MLB Seasons 2000-2020

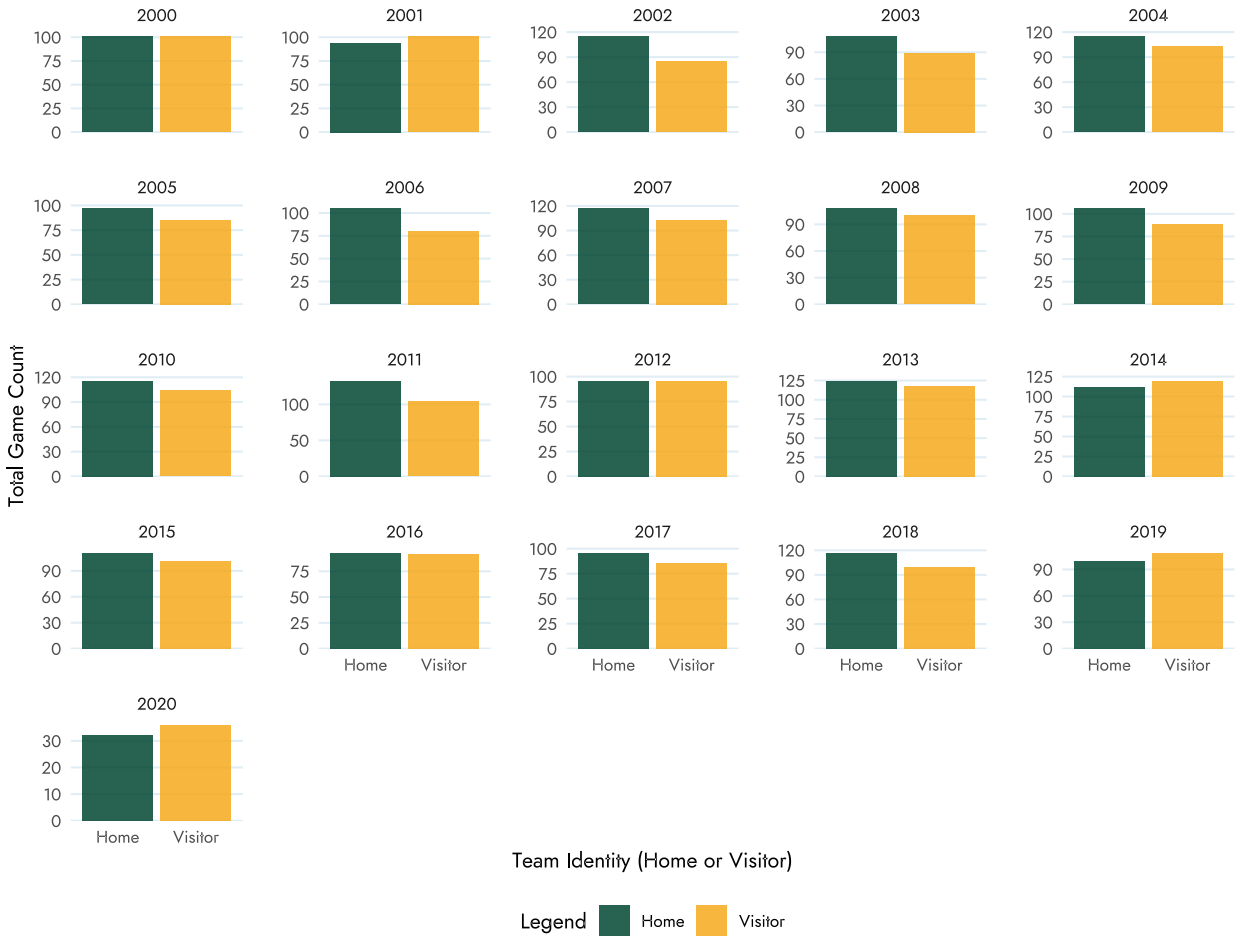

Figure 3: Home vs. Visitor Wins for Extra-Inning Games for MLB Seasons 2000-2020

Table 1: Regular Inning Win Count for Home and Visitor by Season

Home		Visitor	
Wins	Season	Wins	Season
1211	2000	1015	2000
1179	2001	1054	2001
1199	2002	1026	2002
1227	2003	1005	2003
1184	2004	1026	2004
1209	2005	1039	2005
1222	2006	1022	2006
1201	2007	1010	2007
1243	2008	977	2008
1227	2009	1008	2009
1242	2010	968	2010
1143	2011	1049	2011
1199	2012	1039	2012
1182	2013	1006	2013
1176	2014	1022	2014
1205	2015	1012	2015
1194	2016	1048	2016
1215	2017	1033	2017
1166	2018	1049	2018
1187	2019	1034	2019
462	2020	368	2020

Table 2: Extra Inning Win Count for Home and Visitor by Season

Home		Visitor	
Wins	Season	Wins	Season
101	2000	101	2000
94	2001	101	2001
115	2002	85	2002
108	2003	89	2003
115	2004	103	2004
97	2005	85	2005
105	2006	80	2006
117	2007	103	2007
108	2008	100	2008
106	2009	89	2009
116	2010	104	2010
133	2011	104	2011
96	2012	96	2012
125	2013	118	2013
112	2014	120	2014
111	2015	101	2015
93	2016	92	2016
96	2017	86	2017
117	2018	99	2018
99	2019	109	2019
32	2020	36	2020

Analysis

Bayesian Two-Sample t-test

A Bayesian two-sample t-test was performed using the `bayes.t.test()` function from the `Bolstad` package (Curran & Bolstad, 2020) to compare whether the length of extra-innings games in the 2020 MLB season was shortened by the new rule. This was done between the 2020 and 2019 season, the 2020 and 2018 season, and the 2019 and 2018 season. The first comparison was done to compare two seasons in which the rule was implemented and the visitor team won more extra inning games. The second comparison was done to compare two seasons in which the rule was implemented and the visitor won more extra-inning games in only one season. And, the last comparison was done to compare two seasons in which the rule was not implemented and the visitor won more extra-inning games in one of the seasons. Confidence interval was set to 95% for each test.

2020 - 2019 Comparison

The 2020 season (population, N = 68; sample, n = 60) and 2019 season (population, N = 208; sample, n = 60) comparison returned a posterior mean for the 2020 season of M = 61.95 (population mean, M = 61.73) and a posterior mean for the 2019 season of 68.62 (population mean, M = 66.85) (t(118) = -3.81, p = .0002, CI[-10.13, -3.21]). Because of the differences in population sizes (2020 season, N = 68; 2019 season, N = 208), a sample size of 60 (n = 60) was used for both datasets. The results suggest that the implementation of the runner-on-second rule did shorten the length of extra-inning games; however, given the differences in population sizes, the results may be biased due to a small sample size from the 2020 season.

2020 - 2018 Comparison

As an additional analysis, the 2020 season (population, N = 68; sample, n = 60) was also compared using the `bayes.t.test()` function (Curran & Bolstad, 2020) to the 2018 season (population, N = 216; sample, n = 60). Again, due to the differences in population size, a sample size of 60 was used for both datasets. This analysis returned a posterior mean of M = 61.77 for the 2020 season (population mean, M = 61.73), and a posterior mean of M = 67.72 for the 2018 season (population mean, M = 66.23) (t(118) = -4.17, p = .000006, CI[-8.78, -3.13]). Results again suggest that the new rule shortened the length of extra-inning games; however, again the results may be biased due to the size of the 2020 dataset.

2019 - 2018 Comparison

As a final analysis, the 2019 season (population, N = 208; sample, n = 60) was compared to the 2018 season (population, N = 216; sample, n = 60) using the `bayes.t.test()` function (Curran & Bolstad, 2020). The sample size was kept at 60 for this analysis for the purposes of consistency. Results from this test returned a posterior mean of M = 66.03 for the 2019 season (population mean, M = 66.85), and a posterior mean of M = 65.92 for the 2018 season (population mean, M = 66.23) (t(118) = -0.07, p = 9.42, CI[-3.05, 3.29]). Results from this test suggest that there is no difference in the length of extra-inning games between seasons without the runner-on-second rule.

Runs-Expectancy Matrix

To understand the change in probability of a run scoring in a half-inning, a run expectancy (RE24) matrix was generated. In baseball, there are three possible states of outs before an inning is over (i.e., zero, one, or two outs). Additionally, each base (i.e., first, second, and third) can either be in a state of being occupied or not occupied. Thereby giving 24 possible base-out states (8 base states x 3 out states = 24 base-out states), thus the run expectancy matrix is sometimes referred to as the RE24 matrix (Marchi, 2019a). For this analysis, the RE24 was calculated across each season between 2000 to 2020 combined*. Table 4 shows the RE24 matrix for the combined seasons. To read this table, the the first column represents a base state with the next three columns showing an out state. Base states can be read as 0 representing an unoccupied base with 1 representing an occupied base. First base is represented by the first numeral on the left, with second base being the middle numeral, and third base being the right-most numeral. For example, reading the table at the row beginning with 010 means a runner on second. Continuing along this row, it can be seen that in this state with no outs an average number of 1.14 runs are scored, which drops to 0.69 average runs with one out, and 0.33 runs with two outs.

*Note: I did write code that also performs the same calculation for each individual season. This code has been included in this workbook as well in Appendix A below.

Table 4: Runs Expectancy Matrix for All Seasons (2000-2020) Combined

	0 outs	1 out	2 outs
000	0.51	0.27	0.10
001	1.42	0.97	0.37
010	1.14	0.69	0.33
011	2.01	1.41	0.58
100	0.89	0.53	0.23
101	1.80	1.18	0.50
110	1.49	0.92	0.44
111	2.32	1.58	0.77

In addition to the average runs scored, the variance and standard deviation of runs scored from a base-out state were also calculated and are displayed in Table 5. Here it can be seen that the base-out state with the greatest standard deviation is the bases loaded and no one out (111 0), with a variance of $\sigma^2 = 3.34$ and a standard deviation of $\sigma = 1.83$. Additionally, the base-out state with the lowest standard deviation is no one on and two outs (000 2), with a variance of $\sigma^2 = 0.19$ and standard deviation of $\sigma = 0.44$. Notably, the base-out state of runner on second and no one out is towards the middle of the base-out states in terms of standard deviation at position 11 of 24 and with a variance of $\sigma^2 = 1.75$ and standard deviation of $\sigma = 1.32$ (Table 5).

Table 5: Distribution and Central Tendency for Expected Runs for All Seasons (2000-2020) Combined

State	Mean	Median	Variance	SD	Range
000 2	0.1046424	0	0.1932411	0.4395920	0-9
100 2	0.2302677	0	0.4787442	0.6919134	0-9
010 2	0.3271166	0	0.5500225	0.7416350	0-10
000 1	0.2710119	0	0.5524525	0.7432715	0-12
001 2	0.3675043	0	0.5692348	0.7544765	0-11
110 2	0.4414346	0	0.9641316	0.9819020	0-11
101 2	0.5005371	0	1.0014179	1.0007087	0-12
000 0	0.5073049	0	1.0712216	1.0349984	0-14
001 1	0.9651565	1	1.0829098	1.0406295	0-11
100 1	0.5301370	0	1.1275103	1.0618429	0-12
010 1	0.6884575	0	1.1835046	1.0878900	0-12
011 2	0.5810501	0	1.2328890	1.1103553	0-10
001 0	1.4167235	1	1.5863636	1.2595093	0-11
010 0	1.1377177	1	1.7483712	1.3222599	0-14
101 1	1.1820328	1	1.7944072	1.3395549	0-12
111 2	0.7683975	0	1.8166367	1.3478267	0-11
100 0	0.8898628	0	1.8304100	1.3529265	0-14
110 1	0.9208802	0	1.9351109	1.3910827	0-12
011 1	1.4071204	1	2.0222704	1.4220655	0-12
101 0	1.8008865	1	2.3904744	1.5461159	0-13
011 0	2.0092898	2	2.4080953	1.5518039	0-13
110 0	1.4887929	1	2.7390837	1.6550177	0-14
111 1	1.5750796	1	2.7714777	1.6647756	0-11
111 0	2.3238272	2	3.3451838	1.8289844	0-13

However, while these tables show the average expected runs from a given base-out state, there is also the question as to the probability of at least one run scoring given a base-out state. Table 6 shows the mean of value of at least one run scored given each base-out state. Looking at this table, the base-out state with the greatest probability of one or more runs scoring is the bases loaded and no outs (111 0) with a probability of 0.86, and the lowest probability of one or more runs scoring is no one on and two outs (000 2) with a probability of 0.07. Looking to the base-out state of a runner on second and no outs (010 0), or the starting state for each extra-inning game in the 2020 MLB season, it can be seen that at least one run scoring given that state has a probability of 0.62. This is compared to the standard state of beginning of an inning with no runners and no outs (000 0), which has a probability of 0.28 for at least one or more runs scoring.

Table 6: Probability of Scoring at least One Run from a Given Base-Out State

State	Run Probability
111 0	0.8599573
101 0	0.8545246
011 0	0.8530866
001 0	0.8394893
011 1	0.6786357
111 1	0.6611908
001 1	0.6556419
101 1	0.6377960
110 0	0.6198359
010 0	0.6184847
100 0	0.4203736
110 1	0.4132753
010 1	0.4012692
111 2	0.3194076
000 0	0.2766934
101 2	0.2739754
100 1	0.2694704
011 2	0.2596885
001 2	0.2578975
110 2	0.2251555
010 2	0.2174605
000 1	0.1625228
100 2	0.1286415
000 2	0.0708171

State Transitions

A half-inning of baseball lasts until a third out is made and up to that point the game moves between the various possible base-out states. For example, a regulation half-inning could progress from the first batter as follows: no runners and no outs (000 0), a runner on first with no outs (100 0), no runners and two outs (000 2), runner on second and two outs (010 2), to finally three outs and the end of the half-inning. By establishing the current base-out and the subsequent new base-out state, a transition table was produced to calculate the frequency at which one state followed another. From this transition table a proportional table was then produced to view the proportion at which one state moved to another. This table is displayed in Table 7. To read this table, each row and column represents a base-out state, with the values in each cell representing the proportion of moving from the base-out state of that row to the corresponding column base-out state. For example, the first row of the table represents the starting state of a half-inning with no runners and no outs (000 0). Looking along this row, the next most likely state is no runners and one out (000 1) at 0.67. Following that, the next most likely state is runner on first and no outs (100 0) at 0.24. Lastly, any cell with a value of 0 represents a base-out state for which it is not possible for the base-out state of that row to transition. For example, it would not be possible to transition from no runners no outs (000 0) to runner on third and one out (001 1). Looking at the above example, the probability of those series of events would be 0.00085 or 0.085% (0.24 x 0.12 x 0.047 x 0.63).

Table 7: Proportional Matrix for Transitions Between Base-Out States

	000 0	000 1	000 2	001 0	001 1	001 2	010 0	010 1	010 2	011 0	011 1	011 2	100 0	100 1	100 2	101 0	101 1	101 2	110 0	110 1	110 2	111 0	111 1	111 2	3
000 0	0.0306	0.6732	0.0000	0.0057	0.0000	0.0000	0.0512	0.0000	0.0000	0.0000	0.0000	0.0000	0.2393	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
000 1	0.0000	0.0276	0.6808	0.0000	0.0051	0.0000	0.0000	0.0478	0.0000	0.0000	0.0000	0.0000	0.0000	0.2386	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
000 2	0.0000	0.0000	0.0277	0.0000	0.0000	0.0046	0.0000	0.0000	0.0471	0.0000	0.0000	0.0000	0.0000	0.0000	0.2443	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.6763
001 0	0.0245	0.2238	0.0039	0.0062	0.4014	0.0000	0.0506	0.0010	0.0000	0.0001	0.0000	0.0000	0.1726	0.0048	0.0000	0.1111	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
001 1	0.0000	0.0232	0.2101	0.0000	0.0071	0.3480	0.0000	0.0500	0.0056	0.0000	0.0002	0.0000	0.0000	0.1742	0.0232	0.0000	0.1482	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0103
001 2	0.0000	0.0000	0.0223	0.0000	0.0000	0.0049	0.0000	0.0000	0.0433	0.0000	0.0000	0.0000	0.0000	0.0000	0.1308	0.0000	0.0000	0.1645	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.6342
010 0	0.0225	0.0026	0.0063	0.0052	0.2698	0.0000	0.0481	0.3768	0.0000	0.0044	0.0000	0.0000	0.0471	0.0138	0.0000	0.0981	0.0000	0.0000	0.1053	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
010 1	0.0000	0.0244	0.0037	0.0000	0.0058	0.1866	0.0000	0.0514	0.4354	0.0000	0.0032	0.0000	0.0000	0.0575	0.0100	0.0000	0.0685	0.0000	0.0000	0.1457	0.0000	0.0000	0.0000	0.0000	0.0078
010 2	0.0000	0.0000	0.0223	0.0000	0.0000	0.0062	0.0000	0.0000	0.0567	0.0000	0.0000	0.0004	0.0000	0.0000	0.0801	0.0000	0.0000	0.0309	0.0000	0.0000	0.1688	0.0000	0.0000	0.0000	0.6346
011 0	0.0229	0.0028	0.0024	0.0057	0.1519	0.0038	0.0526	0.0897	0.0024	0.0057	0.3550	0.0000	0.0504	0.0029	0.0000	0.0935	0.0082	0.0000	0.0084	0.0068	0.0000	0.1347	0.0000	0.0000	0.0001
011 1	0.0000	0.0183	0.0037	0.0000	0.0060	0.1293	0.0000	0.0483	0.0864	0.0000	0.0054	0.2870	0.0000	0.0542	0.0048	0.0000	0.0772	0.0228	0.0000	0.0069	0.0090	0.0000	0.2277	0.0000	0.0130
011 2	0.0000	0.0000	0.0202	0.0000	0.0000	0.0055	0.0000	0.0000	0.0508	0.0000	0.0000	0.0006	0.0000	0.0000	0.0718	0.0000	0.0000	0.0278	0.0000	0.0000	0.0002	0.0000	0.0000	0.2033	0.6198
100 0	0.0291	0.0005	0.1211	0.0054	0.0026	0.0000	0.0144	0.1127	0.0000	0.0357	0.0000	0.0000	0.0001	0.4341	0.0000	0.0411	0.0000	0.0000	0.2031	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000
100 1	0.0000	0.0293	0.0006	0.0000	0.0061	0.0028	0.0000	0.0158	0.0833	0.0000	0.0330	0.0000	0.0000	0.0002	0.4483	0.0000	0.0439	0.0000	0.0000	0.2073	0.0000	0.0000	0.0000	0.0000	0.1293
100 2	0.0000	0.0000	0.0290	0.0000	0.0000	0.0068	0.0000	0.0000	0.0226	0.0000	0.0000	0.0224	0.0000	0.0000	0.0006	0.0000	0.0000	0.0482	0.0000	0.0000	0.1930	0.0000	0.0000	0.0000	0.6772
101 0	0.0295	0.0009	0.0846	0.0061	0.0025	0.0105	0.0144	0.0488	0.0034	0.0374	0.0460	0.0000	0.0001	0.1720	0.0039	0.0362	0.2508	0.0000	0.1505	0.0224	0.0000	0.0798	0.0000	0.0000	0.0004
101 1	0.0000	0.0270	0.0007	0.0000	0.0065	0.0031	0.0000	0.0157	0.0634	0.0000	0.0364	0.0274	0.0000	0.0002	0.1702	0.0000	0.0403	0.2327	0.0000	0.1424	0.0172	0.0000	0.0831	0.0000	0.1336
101 2	0.0000	0.0000	0.0246	0.0000	0.0000	0.0074	0.0000	0.0000	0.0225	0.0000	0.0000	0.0226	0.0000	0.0000	0.0006	0.0000	0.0000	0.0440	0.0000	0.0000	0.1058	0.0000	0.0000	0.1023	0.6702
110 0	0.0285	0.0005	0.0002	0.0055	0.0012	0.0913	0.0134	0.0025	0.0110	0.0349	0.1265	0.0000	0.0001	0.0023	0.0088	0.0251	0.1020	0.0000	0.0387	0.3391	0.0000	0.1673	0.0000	0.0000	0.0011
110 1	0.0000	0.0283	0.0006	0.0000	0.0062	0.0016	0.0000	0.0162	0.0040	0.0000	0.0356	0.0711	0.0000	0.0002	0.0036	0.0000	0.0302	0.0973	0.0000	0.0482	0.3717	0.0000	0.1553	0.0000	0.1299
110 2	0.0000	0.0000	0.0254	0.0000	0.0000	0.0078	0.0000	0.0000	0.0239	0.0000	0.0000	0.0264	0.0000	0.0000	0.0005	0.0000	0.0000	0.0430	0.0000	0.0000	0.0495	0.0000	0.0000	0.1267	0.6968
111 0	0.0278	0.0006	0.0002	0.0057	0.0022	0.0751	0.0144	0.0034	0.0033	0.0429	0.0419	0.0273	0.0001	0.0027	0.0015	0.0236	0.0993	0.0036	0.0458	0.0695	0.0057	0.1725	0.3303	0.0000	0.0008
111 1	0.0000	0.0274	0.0008	0.0000	0.0072	0.0016	0.0000	0.0148	0.0033	0.0000	0.0405	0.0476	0.0000	0.0001	0.0033	0.0000	0.0270	0.0966	0.0000	0.0478	0.0793	0.0000	0.1591	0.3085	0.1352
111 2	0.0000	0.0000	0.0257	0.0000	0.0000	0.0094	0.0000	0.0000	0.0249	0.0000	0.0000	0.0280	0.0000	0.0000	0.0006	0.0000	0.0000	0.0439	0.0000	0.0000	0.0486	0.0000	0.0000	0.1157	0.7033
	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000

To specifically examine the transition states following runner on second and no outs (010 0), Table 8 was generated to filter for only the transition proportions following that state. Looking at this table it can be seen that the next most likely state following a runner on second and no outs is a runner on second with one out (010 1) at 0.38. This state is followed in likelihood by a runner on third with one out (001 1) at 0.27. However, this table only shows the next most likely base-out state after one at-bat. Alternatively, Table 9 shows the the likelihood of a base-out state after two at-bats, from which it can be seen that the most likely base-out state is runner on third and two outs (001 2) at 0.174. This is additionally followed in likelihood by runner on second and two outs (010 2) at 0.168. Lastly, Table 10 shows the likelihood of base-out states after three at-bats from the beginning state of runner on second and no outs (010 0). In this table it can be seen that the most likely state after three at-bats is three outs at 0.33. This is followed in likelihood by runner on first and two outs at 0.12.

Table 8: Probability of Next Base-Out State After One At-Bat for Man on Second no Outs

state	new_state	prob
010 0	010 1	0.3768
010 0	001 1	0.2698
010 0	110 0	0.1053
010 0	101 0	0.0981
010 0	010 0	0.0481
010 0	100 0	0.0471
010 0	000 0	0.0225
010 0	100 1	0.0138
010 0	000 2	0.0063
010 0	001 0	0.0052
010 0	011 0	0.0044
010 0	000 1	0.0026
010 0	001 2	0.0000
010 0	010 2	0.0000
010 0	011 1	0.0000
010 0	011 2	0.0000
010 0	100 2	0.0000
010 0	101 1	0.0000
010 0	101 2	0.0000
010 0	110 1	0.0000
010 0	110 2	0.0000
010 0	111 0	0.0000
010 0	111 1	0.0000
010 0	111 2	0.0000
010 0	3	0.0000

Table 9: Probability of Next Base-Out State After Two At-Bats for Man on Second no Outs

state	new_state	prob
010 0	001 2	0.1749296
010 0	010 2	0.1685183
010 0	100 1	0.1075513
010 0	101 1	0.1017811
010 0	110 1	0.0956951
010 0	000 2	0.0743900
010 0	010 1	0.0620825
010 0	110 0	0.0335071
010 0	000 1	0.0325420
010 0	111 0	0.0260377
010 0	011 1	0.0211102
010 0	001 1	0.0204255
010 0	100 2	0.0190622
010 0	101 0	0.0138378
010 0	3	0.0119185
010 0	000 0	0.0092645
010 0	011 0	0.0092626
010 0	100 0	0.0087941
010 0	010 0	0.0074621
010 0	001 0	0.0018676
010 0	011 2	0.0000000
010 0	101 2	0.0000000
010 0	110 2	0.0000000
010 0	111 1	0.0000000
010 0	111 2	0.0000000

Table 10: Probability of Next Base-Out State After Three At-Bats for Man on Second no Outs

state	new_state	prob
010 0	3	0.3339878
010 0	100 2	0.1220308
010 0	110 2	0.0697838
010 0	101 2	0.0684726
010 0	010 2	0.0663515
010 0	110 1	0.0641374
010 0	000 2	0.0395434
010 0	111 1	0.0367265
010 0	101 1	0.0301724
010 0	001 2	0.0297609
010 0	100 1	0.0225814
010 0	011 1	0.0202319
010 0	000 1	0.0186301
010 0	010 1	0.0171181
010 0	011 2	0.0168566
010 0	111 0	0.0124492
010 0	110 0	0.0072215
010 0	001 1	0.0070335
010 0	101 0	0.0041235
010 0	100 0	0.0033659
010 0	011 0	0.0032037
010 0	000 0	0.0030522
010 0	010 0	0.0025648
010 0	001 0	0.0006206
010 0	111 2	0.0000000

Runs Simulation

As a final analysis a multi-chain model was generated to simulate the number of runs scored in a half-inning across all seasons. For this analysis, runs scored was simulated using the formula $RUNS = (N_{runners}^{(b)} + O^{(b)} + 1) - (N_{runners}^{(a)} + o^{(a)})$, where runs scored $RUNS$ is equal to difference between the sum of runners $N_{runners}$ and outs O before (b) the event plus one (1) and the number of runners $N_{runners}$ plus outs O after (a) the event (Marchi et al., 2019b). The multi-chain model was generated by computing the number of runs for a given state using the given formula and then samples of size one (1) were selected for the length of a vector of length 25 using the probability weights from the transition matrix. Runs were then returned as an indexed location from the calculated runs as weighted by the probabilities provided by the transition matrix. The simulation was then set to run for 20,000 simulations, results of which can be seen in Table 11. From this table it can be seen that the most likely outcome in a half-inning is that no runs are scored. Additionally, the probability of scoring only one run can be seen as 0.14 (2837/20000), whereas the probability of scoring one or more runs can be seen as 0.27 ((2837+1276+620+286+128+48+27+11+7)/20000). Lastly, the probability of scoring two or more runs in a half-inning is 0.13 ((1276+620+286+128+48+27+11+7)/20000), slightly less than the probability of scoring only one run at 0.14.

Table 11: Multi-Chain Analysis: Possible Runs Scored in Half-Inning Across 20,000 Simulations

runs_simulation	Freq
0	14760
1	2837
2	1276
3	620
4	286
5	128
6	48
7	27
8	11
9	7

Discussion

Given these results, it cannot be concluded that the rule of beginning each extra-inning with a runner on second gives an advantage to the visitor team over the home team anymore than standard extra-inning play. While the rule does increase the expected average number of runs, both teams experience this increase. The disadvantage comes from if the visitor team were to score in extra-inning games, they would only require one run to win the game. Whereas, if the visitor were to score in the top-half of the inning, the home team would require two runs to win the game. Effectively, it is the structure of extra-inning games to begin that gives the visitor an advantage as long as they score in the top-half of the inning. This structure is just exaggerated by the rule of beginning each half-inning with a runner on second by increasing the probability that at least one run will score. However, given that the most likely outcome after three at-bats with a runner on second and no outs is three outs and the end of the half-inning, this advantage may not be so vast. Which may also be explained by other seasons in which the visitor won more extra-inning games than the home team. Additionally, it may also be the case that the structure of extra-inning games lends itself to more randomness and that a similar effect would appear if more games were played under the same structure. Or, that the randomness would become lessened and as more extra-inning games were played a similar effect of regulation length games with the home team winning more than the visitor would appear.

References

Allen, S. (2020, July 13). *A guide to MLB's extra-innings rule for 2020.* The Washington Post. https://www.washingtonpost.com/sports/2020/07/30/mlb-extra-innings-rules-2020/

Baumer, B. (2019, March 15). *baseball_r.* [Code]. GitHub. https://github.com/beanumber/baseball_R/blob/master/scripts/parse_retrosheet_pbp.R

Colin Douglas and Richard Scriven (2020). retrosheet: Import Professional Baseball Data from 'Retrosheet'. R package version 1.1.3. https://CRAN.R-project.org/package=retrosheet

Gatto, T. (2020, August 14). *NHL bubble, explained: A guide to the hub city rules, teams & schedule for Edmonton, Toronto.* Sporting News. https://www.sportingnews.com/us/nhl/news/nhl-bubble-hub-city-rules-teams-schedule-edmonton-toronto/72k8vc0u630k19xalra66xa3c

Haislop, T. (2020, August 26). *NBA bubble explained: A complete guide to the rules, teams, schedule & more for Orlando games.* Sporting News. https://www.sportingnews.com/us/nba/news/nba-bubble-rules-teams-schedule-orlando/zhap66a9hcwq1khmcex3ggabo

Marchi, M., Albert, J., & Baumer, B. S. (2019a). Chapter 5: Value of plays using run expectancy. In, *Analyzing baseball data with R* (2nd ed.), (pp. 111-135). CRC Press

Marchi, M., Albert, J., & Baumer, B. S. (2019b). Chapter 9: Simulation. In, *Analyzing baseball data with R* (2nd ed.), (pp. 201-226). CRC Press

Marchi, M., Albert, J., & Baumer, B. S. (2019c). Appendix A: Retrosheet files reference. In, *Analyzing baseball data with R* (2nd ed.), (pp. 293-301). CRC Press

Max (2013, April 14). *fields* [Data set]. GitHub. https://github.com/maxtoki/baseball_R/blob/master/data/fields.csv

McNamara, A. (2020, July 24). *Toronto Blue Jays to play majority of 2020 home games in Buffalo.* CBS News. https://www.cbsnews.com/news/toronto-blue-jays-home-games-buffalo-2020-season/

Kirill Müller (2020). here: A Simpler Way to Find Your Files. R package version 1.0.1. https://CRAN.R-project.org/package=here

Retrosheet. (2021). *Retrosheet.* https://retrosheet.org/

Wagner, J. (2020, June 24). *Baseball's New Rules: No Spitting, No Arguing, and Lots of Testing.* The New York Times. https://www.nytimes.com/2020/06/24/sports/baseball/mlb-coronavirus-rules.html

H. Wickham. ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York, 2016.

Wickham et al., (2019). Welcome to the tidyverse. Journal of Open Source Software, 4(43), 1686, https://doi.org/10.21105/joss.01686

Appendix A: RMarkdown Code

```
# uncomment any packages that need to be installed

# install tidyverse package
# version 1.3.0
#install.packages("tidyverse")

# install ggplot2 package
# version 3.3.3
#install.packages("ggplot2")

# install showtext package
# version 0.9-2
#install.packages("showtext")

# install here package
# version 1.0.1
#install.packages("here")

# install retrosheet package
# version 1.1.3
#install.packages("retrosheet")
```

```
# loads tidyverse library for working with data
# version 1.3.0
library(tidyverse)

# loads ggplot2 library for creating plots
# version 3.3.3
library(ggplot2)

# loads showtext library for accessing Google fonts
# version 0.9-2
library(showtext)

# load here library for working with directories
# version 1.0.1
library(here)

# load retrosheet package for baseball game data
# version 1.1.3
library(retrosheet)
```

```
# add in fonts from Google fonts using showtext package
# font_add_google function calls the font from Google fonts
font_add_google(name = "Jost", family = "jost-sans-serif")
showtext_auto()
```

```
# read in the read_gl() function from script directory
source(here::here("script/read_gl.R"))
```

```
# read in game log datasets using the read_gl function
read_gl("gl2000_x", "data/gl2000_09/GL2000.TXT")
read_gl("gl2001_x", "data/gl2000_09/GL2001.TXT")
read_gl("gl2002_x", "data/gl2000_09/GL2002.TXT")
read_gl("gl2003_x", "data/gl2000_09/GL2003.TXT")
read_gl("gl2004_x", "data/gl2000_09/GL2004.TXT")
read_gl("gl2005_x", "data/gl2000_09/GL2005.TXT")
read_gl("gl2006_x", "data/gl2000_09/GL2006.TXT")
read_gl("gl2007_x", "data/gl2000_09/GL2007.TXT")
read_gl("gl2008_x", "data/gl2000_09/GL2008.TXT")
read_gl("gl2009_x", "data/gl2000_09/GL2009.TXT")
read_gl("gl2010_x", "data/gl2010_19/GL2010.TXT")
read_gl("gl2011_x", "data/gl2010_19/GL2011.TXT")
read_gl("gl2012_x", "data/gl2010_19/GL2012.TXT")
read_gl("gl2013_x", "data/gl2010_19/GL2013.TXT")
read_gl("gl2014_x", "data/gl2010_19/GL2014.TXT")
read_gl("gl2015_x", "data/gl2010_19/GL2015.TXT")
read_gl("gl2016_x", "data/gl2010_19/GL2016.TXT")
read_gl("gl2017_x", "data/gl2010_19/GL2017.TXT")
read_gl("gl2018_x", "data/gl2010_19/GL2018.TXT")
read_gl("gl2019_x", "data/gl2010_19/GL2019.TXT")
read_gl("gl2020_x", "data/gl2020_20/GL2020.TXT")
```

```
# read in game log datasets using the read_gl function
read_gl("gl2000_r", "data/gl2000_09/GL2000.TXT")
read_gl("gl2001_r", "data/gl2000_09/GL2001.TXT")
read_gl("gl2002_r", "data/gl2000_09/GL2002.TXT")
read_gl("gl2003_r", "data/gl2000_09/GL2003.TXT")
read_gl("gl2004_r", "data/gl2000_09/GL2004.TXT")
read_gl("gl2005_r", "data/gl2000_09/GL2005.TXT")
read_gl("gl2006_r", "data/gl2000_09/GL2006.TXT")
read_gl("gl2007_r", "data/gl2000_09/GL2007.TXT")
read_gl("gl2008_r", "data/gl2000_09/GL2008.TXT")
read_gl("gl2009_r", "data/gl2000_09/GL2009.TXT")
read_gl("gl2010_r", "data/gl2010_19/GL2010.TXT")
read_gl("gl2011_r", "data/gl2010_19/GL2011.TXT")
read_gl("gl2012_r", "data/gl2010_19/GL2012.TXT")
read_gl("gl2013_r", "data/gl2010_19/GL2013.TXT")
read_gl("gl2014_r", "data/gl2010_19/GL2014.TXT")
read_gl("gl2015_r", "data/gl2010_19/GL2015.TXT")
read_gl("gl2016_r", "data/gl2010_19/GL2016.TXT")
read_gl("gl2017_r", "data/gl2010_19/GL2017.TXT")
read_gl("gl2018_r", "data/gl2010_19/GL2018.TXT")
read_gl("gl2019_r", "data/gl2010_19/GL2019.TXT")
read_gl("gl2020_r", "data/gl2020_20/GL2020.TXT")
```

```
# read in the prepare_gl() function from script directory
source(here::here("script/prepare_gl.R"))
```

```
# use the prepare_gl function to prepare a given gamelog
prepare_gl("gl2000_x", "extra")
prepare_gl("gl2001_x", "extra")
prepare_gl("gl2002_x", "extra")
```

```
prepare_gl("gl2003_x", "extra")
prepare_gl("gl2004_x", "extra")
prepare_gl("gl2005_x", "extra")
prepare_gl("gl2006_x", "extra")
prepare_gl("gl2007_x", "extra")
prepare_gl("gl2008_x", "extra")
prepare_gl("gl2009_x", "extra")
prepare_gl("gl2010_x", "extra")
prepare_gl("gl2011_x", "extra")
prepare_gl("gl2012_x", "extra")
prepare_gl("gl2013_x", "extra")
prepare_gl("gl2014_x", "extra")
prepare_gl("gl2015_x", "extra")
prepare_gl("gl2016_x", "extra")
prepare_gl("gl2017_x", "extra")
prepare_gl("gl2018_x", "extra")
prepare_gl("gl2019_x", "extra")
prepare_gl("gl2020_x", "extra")
```

```
# use the prepare_gl function to prepare a given gamelog
prepare_gl("gl2000_r", "regular")
prepare_gl("gl2001_r", "regular")
prepare_gl("gl2002_r", "regular")
prepare_gl("gl2003_r", "regular")
prepare_gl("gl2004_r", "regular")
prepare_gl("gl2005_r", "regular")
prepare_gl("gl2006_r", "regular")
prepare_gl("gl2007_r", "regular")
prepare_gl("gl2008_r", "regular")
prepare_gl("gl2009_r", "regular")
prepare_gl("gl2010_r", "regular")
prepare_gl("gl2011_r", "regular")
prepare_gl("gl2012_r", "regular")
prepare_gl("gl2013_r", "regular")
prepare_gl("gl2014_r", "regular")
prepare_gl("gl2015_r", "regular")
prepare_gl("gl2016_r", "regular")
prepare_gl("gl2017_r", "regular")
prepare_gl("gl2018_r", "regular")
prepare_gl("gl2019_r", "regular")
prepare_gl("gl2020_r", "regular")
```

```
# create temporary holders for data object names ->
# to be removed later
t1 <- ls(pattern = "_r")
t2 <- ls(pattern = "_x")
```

```
# create a list of all regulation dataframes
dfr <- mget(t1)

# create a list of all extras dataframes
dfx <- mget(t2)
```

```r
# bind all extra inning dataframes using the created list
extras <- dplyr::bind_rows(dfx)

# bind all regular inning dataframes using the created list
reg <- dplyr::bind_rows(dfr)
```

```r
# remove the created game log data objects
rm(list = c(t1, t2))
# remove the temporary vectors ->
# remove dfr and dfx logical vectors ->
# and remove game log functions
rm(t1, t2, dfr, dfx, prepare_gl, read_gl)
```

```r
# read in the check_NA() function from script directory
source(here::here("script/check_NA.R"))

# creates dataset of missing value row and column numbers
check_NA(reg, "reg_NA")
check_NA(extras, "extras_NA")

# remove NAs in both datasets
reg <- reg %>%
  filter(!is.na(winning_team))

extras <- extras %>%
  filter(!is.na(winning_team))

# doublecheck NA values
check_NA(reg, "reg_NA")
check_NA(extras, "extras_NA")
```

```r
# extra inning wins/losses totals
# winning team count
extrawin_count <- count(extras, winning_team)
extrawin_count <- extrawin_count %>%
  # rename count column
  rename(count = n) %>%
  # create new column for label
  mutate(game_length = "Extra")

# extra inning wins/losses by year
extra_count_yr <- count(extras, winning_team, year) %>%
  # rename count column
  rename(count = n) %>%
  # keep only unique values
  unique()
```

```r
# regular length wins/losses totals
regwin_count <- count(reg, winning_team)
regwin_count <- regwin_count %>%
  # remove NA values
  filter(winning_team != "NA") %>%
```

```
  # rename count column
  rename(count = n) %>%
  # add label column
  mutate(game_length = "Regulation")

# regular length wins/losses by year
reg_count_yr <- count(reg, winning_team, year) %>%
  # remove NAs
  filter(winning_team != "NA") %>%
  # renamce count column
  rename(count = n) %>%
  # keep only unique values
  unique()
```

```
# combine the counts for regulation and extra inning games
combined_counts <- bind_rows(regwin_count, extrawin_count)
```

```
# generate plot using win counts
ggplot(combined_counts, aes(x = winning_team, y = count, fill = winning_team))+
  # generate bar plot, set alpha
  geom_bar(stat = "identity", alpha = .85)+
  # set plot labels
  labs(x = "Team Identity (Home or Visitor)",
       y = "Total Game Count",
       fill = "Legend")+
  # set plot theme to minimal
  theme_minimal()+
  # set plot font to roboto-slab-serif
  theme(text = element_text(family = "jost-sans-serif"),
        # remove minor grid lines
        panel.grid.minor = element_blank(),
        # remove major x grid lines
        panel.grid.major.x = element_blank(),
        # set y grid lines to blue
        panel.grid.major.y = element_line(colour = "#E0EDF5"),
        # set panel spacing to 10mm
        panel.spacing = unit(10, "mm"))+
  # set colour values
  scale_fill_manual(values = c("#034732", "#f6aa1c"),
                    labels = c("Home", "Visitor"))+
  # set x labels
  scale_x_discrete(labels = c("Home", "Visitor"))+
  # facet wrap by game length
  facet_wrap(vars(game_length),
             scales = "free")
```

```
# generate new plot using regulation inning wins/losses data
ggplot(reg_count_yr, aes(x = winning_team, y = count, fill = winning_team))+
  # generate bar plot
  geom_bar(stat = "identity", alpha = .85)+
  # set plot labels
  labs(x = "Team Identity (Home or Visitor)",
       y = "Total Game Count",
```

```
      fill = "Legend")+
  # set plot theme to minimal
  theme_minimal()+
  # set plot font to roboto-slab-serif
  theme(text = element_text(family = "jost-sans-serif"),
        # remove minor grid lines
        panel.grid.minor = element_blank(),
        # remove major x grid lines
        panel.grid.major.x = element_blank(),
        # set y major grid lines to blue
        panel.grid.major.y = element_line(colour = "#E0EDF5"),
        # set panel spacing to 7mm
        panel.spacing = unit(7, "mm"),
        # set legend position to bottom of plot
        legend.position = "bottom",
        # set legend orientation to horizontal
        legend.direction = "horizontal")+
  # set fill values
  scale_fill_manual(values = c("#034732", "#f6aa1c"),
                    labels = c("Home", "Visitor"))+
  # set x labels
  scale_x_discrete(labels = c("Home", "Visitor"))+
  # set y axis values
  scale_y_continuous(expand = expansion(mult = c(0, 0.05)))+
  # facet wrap by year
  facet_wrap(vars(year),
             nrow = 5,
             scales = "free_y")
```

```
# generate new plot using extra inning wins/losses data
ggplot(extra_count_yr, aes(x = winning_team, y = count, fill = winning_team))+
  # generate bar plot
  geom_bar(stat = "identity", alpha = .85)+
  # set plot labels
  labs(x = "Team Identity (Home or Visitor)",
       y = "Total Game Count",
       fill = "Legend")+
  # set plot theme to minimal
  theme_minimal()+
  # set plot font to roboto-slab-serif
  theme(text = element_text(family = "jost-sans-serif"),
        # remove minor grid lines
        panel.grid.minor = element_blank(),
        # remove major x grid lines
        panel.grid.major.x = element_blank(),
        # set major y grid lines to blue
        panel.grid.major.y = element_line(colour = "#E0EDF5"),
        # set panel spacing to 7mm
        panel.spacing = unit(7, "mm"),
        # set legend position to bottom of plot
        legend.position = "bottom",
        # set legend orientation to horizontal
        legend.direction = "horizontal")+
```

```
  # set fill values
  scale_fill_manual(values = c("#034732", "#f6aa1c"),
                    labels = c("Home", "Visitor"))+
  # set x labels
  scale_x_discrete(labels = c("Home", "Visitor"))+
  # set y axis values
  scale_y_continuous(expand = expansion(mult = c(0, 0.05)))+
  # facet wrap by year
  facet_wrap(vars(year),
             nrow = 5,
             scales = "free_y")
```

```
# set regular inning wins into placeholders ->
# to produce tables in PDF
t1 <- reg_count_yr %>%
  # filter for only home team wins
  dplyr::filter(winning_team == "home")%>%
  # rename columns for table columns
  dplyr::rename("Winning Team" = winning_team,
                "Season" = year,
                "Wins" = count) %>%
  # move Count column before Season column
  dplyr::select(Wins, Season)

t2 <- reg_count_yr %>%
  # filter for only visitor wins
  dplyr::filter(winning_team == "visitor")%>%
  # rename columns for table columns
  dplyr::rename("Winning Team" = winning_team,
                "Season" = year,
                "Wins" = count) %>%
  # select only winning team and count
  dplyr::select(Wins, Season)

# bind placeholders side by side
t3 <- cbind(t1, t2)
# create kable table
kableExtra::kbl(t3,
                caption = "Regular Inning Win Count for Home and Visitor by Season") %>%
  kableExtra::kable_styling(position = "center", latex_options = "HOLD_position") %>%
  kableExtra::add_header_above(c("Home" = 2, "Visitor" = 2))

# remove placeholders
rm(t1, t2, t3)
```

```
# set extra inning wins into placeholders ->
# to produce tables in PDF
t1 <- extra_count_yr %>%
  # filter for only home team wins
  dplyr::filter(winning_team == "home") %>%
  # rename columns for table columns
  dplyr::rename("Winning Team" = winning_team,
                "Season" = year,
```

```
                "Wins" = count) %>%
  # move Count column before Season column
  dplyr::select(Wins, Season)

t2 <- extra_count_yr %>%
  # filter for only visitor wins
  dplyr::filter(winning_team == "visitor") %>%
  # rename columns for table columns
  dplyr::rename("Winning Team" = winning_team,
                "Season" = year,
                "Wins" = count) %>%
  # select only winning team and count
  dplyr::select(Wins, Season)

# bind placeholders side by side
t3 <- cbind(t1, t2)

# create kable table
kableExtra::kbl(t3,
                caption = "Extra Inning Win Count for Home and Visitor by Season") %>%
  kableExtra::kable_styling(position = "center", latex_options = "HOLD_position") %>%
  kableExtra::add_header_above(c("Home" = 2, "Visitor" = 2))

# remove placeholders
rm(t1, t2, t3)
```

```
# filter extra inning games into 2020, 2019, and 2018 seasons
extras_20 <- extras %>%
  dplyr::filter(year == 2020)

extras_19 <- extras %>%
  dplyr::filter(year == 2019)

extras_18 <- extras %>%
  dplyr::filter(year == 2018)

##----
# Mean game lengths
mean(extras_20$length_of_game_outs)
mean(extras_19$length_of_game_outs)
mean(extras_18$length_of_game_outs)

##----
# Bayes t.test

# use bayes.t.test function from Bolstad package to calculate two-samples t-test for ->
# 2020 2019 mean length of extra-innings games
bt_2019 <- Bolstad::bayes.t.test(sample(extras_20$length_of_game_outs, size = 60), sample(extras_19$leng
# 2020 2018 mean length of extra-innings games
bt_2018 <- Bolstad::bayes.t.test(sample(extras_20$length_of_game_outs, size = 60), sample(extras_18$leng
# 2019 2018 mean length of extra-innings games
bt_1918 <- Bolstad::bayes.t.test(sample(extras_19$length_of_game_outs, size = 60), sample(extras_18$leng
```

```
##----
# FIELDS DATA

# read in fields data ->
# data is from https://github.com/maxtoki/baseball_R/tree/master/data
fields <- read_csv(here::here("data/fields.csv"))
```

```
# code in this section has been co-opted from chapter nine ->
# of Analyzing Baseball Data with R by Marchi, Albert, and Baumer (2018).

##----
# LOAD FANGRAPHS DATA

# create temporary placeholder to hold list of all*.csv files
t1 <- list.files(path = here::here("data"), pattern = ".*all.*\\.csv$")
# create temporary placeholder for df renames
t2 <- c("df2000", "df2001", "df2002", "df2003", "df2004", "df2005",
        "df2006", "df2007", "df2008", "df2009", "df2010", "df2011",
        "df2012", "df2013", "df2014", "df2015", "df2016", "df2017",
        "df2018", "df2019", "df2020")

# for the length of the temporary vector
for(i in 1:length(t1)){
  # assign the matching character string from t2 ->
  # as a dataframe read in using read_csv() from readr ->
  # pull in column names from fields data and Header ->
  # set NA to character
  assign(t2[i], readr::read_csv(here::here("data", t1[i]),
                          col_names = pull(fields, Header),
                          na = character()) %>%
           dplyr::rename_all(tolower) %>%
           # mutate new column runs ->
           # new column half_inning ->
           # and new column runs_scored, using mutate() from dplyr
           dplyr::mutate(runs = away_score_ct + home_score_ct,
                  half_inning = paste(game_id, inn_ct, bat_home_id),
                  runs_scored = (bat_dest_id > 3) + (run1_dest_id > 3) +
                    (run2_dest_id > 3) + (run3_dest_id > 3),
                  year = substr(game_id, 4, 7)))
}
```

```
##----
# CREATE COMBINED DATASET
# creates a dataset of 3,915,193 observations at 1.9GB

# bind all 20 seasons into one dataset by row
dftotal <- rbind(df2000, df2001, df2002, df2003, df2004, df2005, df2006, df2007, df2008,
                 df2009, df2010, df2011, df2012, df2013, df2014, df2015, df2016, df2017,
                 df2018, df2019, df2020)

# check for NA values in all datasets
#check_NA(dftotal, "dftotal_NA")
```

```
# filter for only column numbers containing NA values
#dftotal_NA %>%
#  select(col) %>%
#  unique()

# remove dftotal NA locations object
#rm(dftotal_NA)
```

```
##----
# GENERATE HALF-INNING DATA

# for the length of the temporary vector
for(i in 1:length(t2)){
  # assign to a new data object with suffix _hi
  assign(paste0(t2[i], "_hi"),
         # group by the created half_inning column
         dplyr::group_by(get(t2[i]), half_inning) %>%
         # summarize outs_inning as sum of event_outs_ct
         dplyr::summarize(outs_inning = sum(event_outs_ct),
                   # summarize runs_inning as a sum of runs_scored
                   runs_inning = sum(runs_scored),
                   # summarize runs_start as the first value of runs
                   runs_start = dplyr::first(runs),
                   # summarize max_runs as runs_inning + runs_start
                   max_runs = runs_inning + runs_start))
}

##----
# JOIN HALF-INNINGS DATA TO FULL DATA

# join half_inning data to full season data ->
# for the length of t2
for(i in 1:length(t2)){
  # assign the name at the index location in t2 ->
  # join the object at the index location in t2 with object in t3 ->
  # join by half_inning column
  assign(paste0(t2[i], "_roi"), dplyr::inner_join(get(t2[i]), get(paste0(t2[i], "_hi")),
                                                  by = "half_inning") %>%
           # mutate new column runs in the remainder of inning as ->
           # maximum runs - runs
           dplyr::mutate(runs_roi = max_runs - runs) %>%
           # mutate new column to hold binary for yes or no if player on base
           dplyr::mutate(bases = paste(ifelse(base1_run_id > '', 1, 0),
                                       ifelse(base2_run_id > '', 1, 0),
                                       ifelse(base3_run_id > '', 1, 0), sep = ""),
                  # create new column to hold current base-out state
                  state = paste(bases, outs_ct),
                  # create new columns to hold destinations for runners
                  nrunner1 = as.numeric(run1_dest_id == 1 | bat_dest_id == 1),
                  nrunner2 = as.numeric(run1_dest_id == 2 | run2_dest_id == 2 |
                                        bat_dest_id == 2),
                  nrunner3 = as.numeric(run1_dest_id == 3 | run2_dest_id == 3 |
                                        run3_dest_id == 3 | bat_dest_id == 3),
```

```
                  # create new column to hold number of outs
                  nouts = outs_ct + event_outs_ct,
                  # create new column to hold new base state
                  new_bases = paste(nrunner1, nrunner2,
                                    nrunner3, sep = ""),
                  # create new column to hold new base-out state
                  new_state = paste(new_bases, nouts)) %>%
           # filter for only when state does not equal new state or runs are above 0
           dplyr::filter((state != new_state) | (runs_scored > 0)))
}



##----
# GENERATE PROBABILITY RUN EXPECTANCY CHART FOR AT LEAST 1 RUN

# for the length of t2
for(i in 1:length(t2)){
  # assign a new object with indexed name and suffix _prob ->
  # filter for only 3 out innings
  assign(paste0(t2[i], "_prob1"), dplyr::filter(get(paste0(t2[i], "_roi")),
                                                outs_inning == 3) %>%
           # group by state values
           dplyr::group_by(state) %>%
           # calculate the probability of a run given a state as ->
           # the mean of runs_roi when greater than or equal to 1 ->
           # or when at least one run was scored
           dplyr::summarise(run_prob = mean(runs_roi >= 1)))
}



##----
# GENERATE PROBABILITY RUN EXPECTANCY CHART FOR AT LEAST 2 RUNS

# for the length of t2
for(i in 1:length(t2)){
  # assign a new object with indexed name and suffix _prob ->
  # filter for only 3 out innings
  assign(paste0(t2[i], "_prob2"), dplyr::filter(get(paste0(t2[i], "_roi")),
                                                outs_inning == 3) %>%
           # group by state values
           dplyr::group_by(state) %>%
           # calculate the probability of a run given a state as ->
           # the mean of runs_roi when greater than or equal to 1 ->
           # or when at least one run was scored
           dplyr::summarise(run_prob = mean(runs_roi >= 2)))
}



##----
# FILTER DATA FOR INNINGS THAT ONLY REACH 3 OUTS
```

```r
# for the length of t2
for(i in 1:length(t2)){
  # assign new object with indexed t2 name and suffix _ematrix ->
  # filtered from _roi objects for complete innings (outs == 3)
  assign(paste0(t2[i], "_ematrix"), dplyr::filter(get(paste0(t2[i], "_roi")),
                                                  outs_inning == 3))
}



##----
# GENERATE THE MEAN RUNS ROI AND OUTS DATA

# for the length of t2
for(i in 1:length(t2)){
  # assign new objects with suffix _runs ->
  # grouped by state values in _ematrix objects ->
  assign(paste0(t2[i], "_runs"), dplyr::group_by(get(paste0(t2[i], "_ematrix")),
                                                 state) %>%
           # use summarise to compute the mean value of runs_roi values ->
           dplyr::summarise(avg = mean(runs_roi)) %>%
           # use mutate to create new outs column ->
           # using substrings of state starting and stopping at position 5 ->
           dplyr::mutate(outs = substr(state, 5, 5)) %>%
           # arrange new object by outs values
           dplyr::arrange(outs))
}




##----
# GENERATE COMPLETE RUNS EXPECTANCY 24 MATRIX

# for the length of t2
for(i in 1:length(t2)){
  # assign a new object with name of indexed item in t2 ->
  # with suffix _runsout and ->
  # assign new object as a matrix and round values in column 2
  assign(paste0(t2[i], "_re24"), as.matrix(round(get(paste0(t2[i], "_runs"))[2], 2)))
  # assign object under previously used name as a matrix of dimensions 8 rows, 3 columns
  assign(paste0(t2[i], "_re24"), matrix(get(paste0(t2[i], "_re24")), 8, 3))
  # assign object under previously used name and set matrix column names with ->
  # magrittr set_colnames and set matrix row names using magritter set_rownames
  assign(paste0(t2[i], "_re24"), magrittr::set_colnames(get(paste0(t2[i], "_re24")),
                                                        c("0 outs",
                                                          "1 out",
                                                          "2 outs")) %>%
           magrittr::set_rownames(c("000", "001", "010", "011",
                                    "100", "101", "110", "111")))

}
```

```
##----
# GENERATE NEXT TEMPORARY HOLDER

# create empty temporary holder
t3 <- c()

# for the length of t2
# paste _roi to the end of the index object
for(i in 1:length(t2)){
  t3[i] =  paste0(t2[i], "_roi")
}

rm(list = t3)

# for the length of t2
# paste _runs to the end of the index object
for(i in 1:length(t2)){
  t3[i] =  paste0(t2[i], "_runs")
}

rm(list = t3)

# for the length of t2
# paste _ematrix to the end of the index object
for(i in 1:length(t2)){
  t3[i] =  paste0(t2[i], "_ematrix")
}

# remove items in t3
rm(list = t3)

# remove t3
rm(t3)
```

```
# SIMULATION MATRICES

# only for use on laptop
#memory.limit(size = 16000)


##----
# JOIN HALF-INNING DATA TO FULL DATASETS

# join half_inning data to full season data ->
# for the length of t2
for(i in 1:length(t2)){
  # assign the name at the index location in t2 ->
  # join the object at the index location in t2 with object in t3 ->
  # join by half_inning column
  assign(paste0(t2[i], "_prep"), dplyr::inner_join(get(t2[i]), get(paste0(t2[i], "_hi")),
                                                   by = "half_inning") %>%
           # mutate new column runs in the remainder of inning as ->
```

```
            # maxmimum runs - runs
            dplyr::mutate(bases = paste(ifelse(base1_run_id > '', 1, 0),
                                  ifelse(base2_run_id > '', 1, 0),
                                  ifelse(base3_run_id > '', 1, 0), sep = ""),
                   # create new column to hold base-out state
                   state = paste(bases, outs_ct),
                   # create new columns to hold number of runners from runner destinations
                   nrunner1 = as.numeric(run1_dest_id == 1 | bat_dest_id == 1),
                   nrunner2 = as.numeric(run1_dest_id == 2 | run2_dest_id == 2 |
                                         bat_dest_id == 2),
                   nrunner3 = as.numeric(run1_dest_id == 3 | run2_dest_id == 3 |
                                         run3_dest_id == 3 | bat_dest_id == 3),
                   # create new column to hold new outs count
                   nouts = outs_ct + event_outs_ct,
                   # create new column to hold new base state
                   new_bases = paste(nrunner1, nrunner2,
                                     nrunner3, sep = ""),
                   # create new column to hold new base-out state
                   new_state = paste(new_bases, nouts)) %>%
            # filter for when state does not equal new state or runs above 0
            dplyr::filter((state != new_state) | (runs_scored > 0)) %>%
            # filter for when for only complete innings and for only batter events
            dplyr::filter(outs_inning == 3, bat_event_fl == T) %>%
            # create new state column to replace 3rd out state
            dplyr::mutate(new_state = gsub("[0-1]{3} 3", "3", new_state)))
}




##----
# GENERATE TRANSITION MATRICES

# create full transition matrices
for(i in 1:length(t2)){
  # assign new object with suffix tmatrix
  assign(paste0(t2[i], "_tmatrix"),
         # select only state and new_state columns
         dplyr::select(get(paste0(t2[i], "_prep")),
                       state,
                       new_state) %>%
           # assign as table
           table())
}




##----
# GENERATE PROBABILITY TABLES FROM TRANSITION MATRICES

# for the length of chr t3
# assign a new probability table matrix with suffix _pmatrix
```

```
# then assign a new data object of the same name and bind rows from ->
# the probability table matrix
for(i in 1:length(t2)){
  assign(paste0(t2[i], "_pmatrix"), prop.table(get(paste0(t2[i], "_tmatrix")), 1))
  assign(paste0(t2[i], "_pmatrix"), rbind(get(paste0(t2[i], "_pmatrix")),
                                          c(rep(0, 24), 1)))
}



##----
# GENERATE NEXT TEMPORARY HOLDER

# create empty temporary holder
t3 <- c()

# for the length of t2
# paste _hi to the end of the index object
for(i in 1:length(t2)){
  t3[i] =  paste0(t2[i], "_hi")
}

# remove objects in t3
rm(list = t3)

# for the length of t2
# paste _prep to the end of the index object
for(i in 1:length(t2)){
  t3[i] =  paste0(t2[i], "_prep")
}

# remove objects in t3
rm(list = t3)

# remove t3
rm(t3)
```

```
##----
# CREATE HALF-INNING DATA

# assign new data object for half-inning data
dftotal_hi <- dftotal %>%
  # group by half-inning values
  dplyr::group_by(half_inning) %>%
  # summarise values ->
  # outs inning is equal to event outs count
  dplyr::summarise(outs_inning = sum(event_outs_ct),
                   # runs inning is equal to sum of runs scored
                   runs_inning = sum(runs_scored),
                   # runs at start is equal to the first runs
                   runs_start = dplyr::first(runs),
                   # maximum runs is runs inning + runs start
                   max_runs = runs_inning + runs_start)
```

```
##----
# JOIN DATA

# assign new data object
dftotal_joined <- dftotal %>%
  # use inner join to join half-inning data to original data
  dplyr::inner_join(dftotal_hi, by = "half_inning") %>%
  # create new column runs_roi that is difference of max runs and runs
  dplyr::mutate(runs_roi = max_runs - runs) %>%
  # create new column bases that sets current base state ->
  # if base1 is blank set to 0, if else set to 1 ->
  # run for each base
  dplyr::mutate(bases = paste(ifelse(base1_run_id > '', 1, 0),
                              ifelse(base2_run_id > '', 1, 0),
                              ifelse(base3_run_id > '', 1, 0), sep = ""),
                # create new column state that is base state and current out count
                state = paste(bases, outs_ct),
                # create new columns to track number of runners and positions
                nrunner1 = as.numeric(run1_dest_id == 1 | bat_dest_id == 1),
                nrunner2 = as.numeric(run1_dest_id == 2 | run2_dest_id == 2 |
                                      bat_dest_id == 2),
                nrunner3 = as.numeric(run1_dest_id == 3 | run2_dest_id == 3 |
                                      run3_dest_id == 3 | bat_dest_id == 3),
                # create new column to count number of outs
                nouts = outs_ct + event_outs_ct,
                # create new column to track new base state
                new_bases = paste(nrunner1, nrunner2, nrunner3, sep = ""),
                # create new column that holds new state with->
                # base position and number of outs
                new_state = paste(new_bases, nouts)) %>%
  # filter such that state does not equal the new state or runs is greater than 0
  dplyr::filter((state != new_state) | (runs_scored > 0))


##----
# CALCULATE RUN PROBABILITY

# assign new data object
dftotal_prob1 <- dftotal_joined %>%
  # filter for only full innings
  dplyr::filter(outs_inning == 3) %>%
  # group by state
  dplyr::group_by(state) %>%
  # suummarise run probability of scoring at least one run from state
  dplyr::summarise(run_prob = mean(runs_roi >= 1)) %>%
  dplyr::arrange(desc(run_prob)) %>%
  dplyr::rename(State = state, `Run Probability` = run_prob)


##----
# BUILD ESTIMATION MATRIX

# reassign data object
```

```r
dftotal_summarise <- dftotal_joined %>%
  # filter for only complete innings
  dplyr::filter(outs_inning == 3) %>%
  # group by state
  dplyr::group_by(state) %>%
  # summarise to calculate the mean/average runs given a state and outs
  dplyr::summarise(`mean` = mean(runs_roi),
                   `median` = median(runs_roi),
                   variance = var(runs_roi),
                   `sd` = sd(runs_roi),
                   `range` = paste(min(runs_roi), "-", max(runs_roi), sep = "")) %>%
  # create new column to hold outs count
  dplyr::mutate(outs = substr(state, 5, 5)) %>%
  # arrange by outs count
  dplyr::arrange(outs)



##----
# RE24 MATRIX

# assign new object as an 8X3 matrix ->
# round values to 2nd decimal place
re24total <- matrix(round(dftotal_summarise$mean, 2), 8, 3) %>%
  # set column names to given vector values
  magrittr::set_colnames(c("0 outs", "1 out", "2 outs")) %>%
  # set row names to given vector values
  magrittr::set_rownames(c("000", "001", "010", "011", "100", "101", "110", "111"))
```

```r
# generate kable table for total RE24 matrix
kableExtra::kbl(re24total,
                caption = "Runs Expectancy Matrix for All Seasons (2000-2020) Combined"
                ) %>%
  kableExtra::kable_styling(position = "center", latex_options = "HOLD_position")
```

```r
# change dataset column names to title format ->
# makes the table oh so pretty
names(dftotal_summarise) <- stringr::str_to_title(names(dftotal_summarise))

# using summarise dataset
dftotal_summarise %>%
  # remove outs column
  dplyr::select(-Outs) %>%
  # fully capitalize SD
  dplyr::rename(SD = Sd) %>%
  # arrange the table by SD
  dplyr::arrange(SD) %>%
  # generate kable table
  kableExtra::kbl(caption = "Distribution and Central Tendency for Expected Runs for All
                  Seasons (2000-2020) Combined") %>%
  kableExtra::kable_styling(position = "center", latex_options = "HOLD_position")
```

```
kableExtra::kbl(dftotal_prob1, caption = "Probability of Scoring at least One Run from a
                Given Base-Out State") %>%
  kableExtra::kable_styling(position = "center", latex_options = "HOLD_position")
```

```
##----
# JOIN DATA

# assign new data object
dftotal_joined <- dftotal %>%
  # join half-inning data to original data
  dplyr::inner_join(dftotal_hi, by = "half_inning") %>%
  # create new column runs_roi that is difference of max runs and runs
  dplyr::mutate(runs_roi = max_runs - runs) %>%
  # create new column bases that sets current base state ->
  # if base1 is blank set to 0, if else set to 1 ->
  # run for each base
  dplyr::mutate(bases = paste(ifelse(base1_run_id > '', 1, 0),
                              ifelse(base2_run_id > '', 1, 0),
                              ifelse(base3_run_id > '', 1, 0), sep = ""),
                # create new column state that is base state and current out count
                state = paste(bases, outs_ct),
                # create new columns to track number of runners and positions
                nrunner1 = as.numeric(run1_dest_id == 1 | bat_dest_id == 1),
                nrunner2 = as.numeric(run1_dest_id == 2 | run2_dest_id == 2 |
                                      bat_dest_id == 2),
                nrunner3 = as.numeric(run1_dest_id == 3 | run2_dest_id == 3 |
                                      run3_dest_id == 3 | bat_dest_id == 3),
                # create new column to count number of outs
                nouts = outs_ct + event_outs_ct,
                # create new column to track new base state
                new_bases = paste(nrunner1, nrunner2, nrunner3, sep = ""),
                # create new column that holds new state with->
                # base position and number of outs
                new_state = paste(new_bases, nouts)) %>%
  # filter such that state does not equal the new state or runs is greater than 0
  dplyr::filter((state != new_state) | (runs_scored > 0)) %>%
  # filter for complete innings and that the event was a batting event
  dplyr::filter(outs_inning == 3, bat_event_fl == T) %>%
  # create new column to hold new state
  dplyr::mutate(new_state = gsub("[0-1]{3} 3", "3", new_state))

##----
# GENERATE TRANSITION MATRIX

# assign new value object
tmatrix_total <- dftotal_joined %>%
  # select only state and new state columns
  select(state, new_state) %>%
  # assign object as table
  table()

##----
# GENERATE PROP TABLE
```

```r
# create prop table object from transition matrix
pmatrix_total <- round(prop.table(tmatrix_total, 1), 4)
# bind final column for 3 out state
pmatrix_total <- rbind(pmatrix_total, c(rep(0, 24), 1))
```

```r
kableExtra::kbl(pmatrix_total, caption = "Proportional Matrix for Transitions Between
                Base-Out States") %>%
  kableExtra::kable_styling(position = "center", latex_options = c("HOLD_position",
                                                                   "striped",
                                                                   "scale_down")) %>%
  kableExtra::landscape()
```

```r
# multiply probability matrix by itself three times ->
# finds probabilities of events after one at-bat
p1 <- pmatrix_total

# pipe multiplied probabilities
p1 <- p1 %>%
  # coerce object as tibble with rownames as state
  tibble::as_tibble(rownames = "state") %>%
  # filter for man on second no outs
  dplyr::filter(state == "010 0") %>%
  # gather prob values using new_state as key
  tidyr::gather(key = "new_state", value = "prob", -state) %>%
  # arrange by descending probability
  dplyr::arrange(desc(prob))

kableExtra::kbl(p1, caption = "Probability of Next Base-Out State After One At-Bat for
                Man on Second no Outs") %>%
  kableExtra::kable_styling(position = "center", latex_options = "HOLD_position")
```

```r
# multiply probability matrix by itself three times ->
# finds probabilities of events after two at-bats
p2 <- pmatrix_total %*% pmatrix_total

# pipe multiplied probabilities
p2 <- p2 %>%
  # coerce object as tibble with rownames as state
  tibble::as_tibble(rownames = "state") %>%
  # filter for man on second no outs
  dplyr::filter(state == "010 0") %>%
  # gather prob values using new_state as key
  tidyr::gather(key = "new_state", value = "prob", -state) %>%
  # arrange by descending probability
  dplyr::arrange(desc(prob))

kableExtra::kbl(p2, caption = "Probability of Next Base-Out State After Two At-Bats for
                Man on Second no Outs") %>%
  kableExtra::kable_styling(position = "center", latex_options = "HOLD_position")
```

```r
# multiply probability matrix by itself three times ->
# finds probabilities of events after three at-bats
p3 <- pmatrix_total %*% pmatrix_total %*% pmatrix_total

# pipe multiplied probabilities
p3 <- p3 %>%
  # coerce object as tibble with rownames as state
  tibble::as_tibble(rownames = "state") %>%
  # filter for man on second no outs
  dplyr::filter(state == "010 0") %>%
  # gather prob values using new_state as key
  tidyr::gather(key = "new_state", value = "prob", -state) %>%
  # arrange by descending probability
  dplyr::arrange(desc(prob))

kableExtra::kbl(p3, caption = "Probability of Next Base-Out State After Three At-Bats
                for Man on Second no Outs") %>%
  kableExtra::kable_styling(position = "center", latex_options = "HOLD_position")
```

```r
# assign new function count runners out ->
# takes one argument, a state ->
# returns the sum of the number of runners and outs
count_runners_out <- function(.state){
  # pipe given state value
  .state %>%
    # look for pattern ""
    stringr::str_split("") %>%
    # pluck value at index 1
    purrr::pluck(1) %>%
    # set as numeric
    as.numeric() %>%
    # sum values with removal of NAs ->
    # gives number of runners and outs
    sum(na.rm = T)
}
```

```r
# for the length of t2
for(i in 1:length(t2)){
  # assign a new object with suffix _rout and apply count_runners_out function to ->
  # object _tmatrix
  assign(paste0(t2[i], "_rout"), sapply(row.names(get(paste0(t2[1], "_tmatrix"))),
         count_runners_out))[-25]
}

t3 <- c()

# reuse third temporary holder for "_pmatrix" table names
# for the length of t2
# paste _pmatrix to the end of the index object
for(i in 1:length(t2)){
  t3[i] =  paste0(t2[i], "_rmatrix")
}
```

```r
# create runs calculation matrix ->
# for the length of t2 ->
for(i in 1:length(t2)){
  # assign a new object with suffix _rmatrix ->
  # use the outer function to perform the run calculation shown in formula ->
  # runs = nrunnersbefore + outs before + 1 - nrunnersafter + outsafter
  assign(paste0(t2[i], "_rmatrix"), outer(get(paste0(t2[i], "_rout")) + 1,
                                          get(paste0(t2[i], "_rout")), FUN = "-"))
  # assign names in tmatrix to rmatrix
  names(t3[1]) <- names(paste0(t2[i], "_tmatrix"))[-25]
  # reassign new rmatrix objects and bind new column
  assign(paste0(t2[i], "_rmatrix"), cbind(get(paste0(t2[i], "_rmatrix")), rep(0, 24)))
}
```

```r
# assign new object runners_out using sapply to apply ->
# count_runners_out function to tmatrix ->
# dropping index 25 as this is where 000 0 repeats
runners_out <- sapply(row.names(tmatrix_total),
                      count_runners_out)[-25]

# assign new object rtotal to calculate runs scored formula
rtotal <- outer(runners_out + 1, runners_out, FUN = "-")
# assign names in rtotal using names in tmatrix
names(rtotal) <- names(tmatrix_total)[-25]
# bind final column of 0s to rtotal
rtotal <- cbind(rtotal, rep(0, 24))
```

```r
# assign new function to simulate runs in half inning ->
# takes tmatrix and rmatrix as arguments ->
# set start to begin at 1
sim_half_inning <- function(.tmatrix, .rmatrix, start = 1){
  # assign start number to s
  s <- start
  # set path to NULL
  path <- NULL
  # set runs to start at 0
  runs <- 0
  # while s is less than 25
  while(s < 25){
    # assign to s_new from 1 through 25 of size 1 ->
    # with probability from given tmatrix at position s
    s_new <- sample(1:25, size = 1, prob = .tmatrix[s, ])
    # assign path as path and sample
    path <- c(path, s_new)
    # assign runs as runs + run matrix at position s and s_new
    runs <- runs + .rmatrix[s, s_new]
    # assign s as s_new
    s <- s_new
  }
  # return runs
  runs
}
```

```r
# set randomization seed
# lucky number and hockey jersey number
set.seed(515)

# run 20,000 simulations
runs_simulation <- replicate(20000, sim_half_inning(tmatrix_total, rtotal))

# assign simulations as a table
runs_table <- table(runs_simulation)

# calculate the probability of scoring at least 1 run
prob_one <- sum(runs_simulation == 1) / 20000

# calculate the probability of scoring 1 or more runs
prob_one_or_more <- sum(runs_simulation >= 1) / 20000

# calculate the probability of scoring 1 or more runs
prob_two_or_more <- sum(runs_simulation >= 2) / 20000
```

```r
kableExtra::kbl(runs_table, caption = "Multi-Chain Analysis: Possible Runs Scored in
                Half-Inning Across 20,000 Simulations") %>%
  kableExtra::kable_styling(position = "center", latex_options = "HOLD_position")
```

Appendix B: Functions

```
##----
#CHECK_NA FUNCTION

# function to identify missing values by column and row
# takes two parameters .data, a dataset, and .var, a character string
check_NA <- function(.data, .var){
  # returns an array index for missing values
  assign(.var, which(is.na(.data), arr.ind = TRUE), envir = .GlobalEnv)
  # creates dataframe with given character string
  assign(.var, as.data.frame(get(.var)), envir = .GlobalEnv)
  # removes duplicate values
  assign(.var, unique(get(.var)), envir = .GlobalEnv)
}

##----
# READ GAME LOG FUNCTION

# install tidyverse package
# version 1.3.0
#install.packages("tidyverse")

# loads tidyverse library for working with data
# version 1.3.0
library(tidyverse)


# read_gl function
# reads in a game log dataset from a given directory and creates a data object using a given name ->
# uses the here() function from the here package to read data directory
# using the rename function() from dplyr package ->
# renames columns to something more representative and understandable
# using mutate() function from dplyr ->
# adds a column with the year of the season
read_gl <- function(.data, file_dir){
  assign(.data, read.table(here::here(file_dir), header = FALSE, sep = ","), envir = .GlobalEnv)
  assign(.data, dplyr::rename(get(.data), date = V1, game_type = V2, weekday = V3, visiting_team = V4, v
                              visiting_game_no = V6, home_team = V7, home_league = V8, home_game_no = V9
                              home_score = V11, length_of_game_outs = V12, day_night = V13, completion =
                              protest = V16, parkid = V17, attendance = V18, time_of_game_mins = V19, vi
                              home_scoreline = V21, visiting_ab = V22, visiting_hits = V23, visiting_dbl
                              visiting_hrs = V26, visiting_rbi = V27, visiting_sachit = V28, visiting_sa
                              visiting_bb = V31, visiting_ibb = V32, visiting_k = V33, visiting_sb = V34
                              visiting_odp = V36, visiting_ci = V37, visiting_lob = V38, visiting_pitche
                              visiting_ter = V41, visiting_wp = V42, visiting_balks = V43, visiting_putc
                              visiting_errors = V46, visiting_passedballs = V47, visiting_ddp = V48, vis
                              home_hits = V51, home_dbls = V52, home_trpls = V53, home_hrs = V54, home_
                              home_sacfly = V57, home_hbp = V58, home_bb = V59, home_ibb = V60, home_k =
                              home_odp = V64, home_ci = V65, home_lob = V66, home_pitcher_no = V67, home
                              home_wp = V70, home_balks = V71, home_putouts = V72, home_assists = V73, h
                              home_passedballs = V75, home_ddp = V76, home_dtp = V77, umphome_id = V78,
                              ump1b_name = V81, ump2b_id = V82, ump2b_name = V83, ump3b_id = V84, ump3b_
```

```
                                umplf_name = V87, umprf_id = V88, umprf_name = V89, visiting_mngrid = V90
                                home_mngrid = V92, home_mngrname = V93, win_pitchid = V94, win_pitchname =
                                lose_pitchname = V97, sv_pitchid = V98, sv_pitchname = V99, game_winning_
                                game_winning_rbi_name = V101, vis_strtp_id = V102, vis_strtp_name = V103,
                                home_strtp_name = V105, vis_strt1_id = V106, vis_strt1_name = V107, vis_s
                                vis_strt2_name = V110, vis_strt2_def = V111, vis_strt3_id = V112, vis_str
                                vis_strt4_id = V115, vis_strt4_name = V116, vis_strt4_def = V117, vis_str
                                vis_strt5_def = V120, vis_strt6_id = V124, vis_strt6_name = V122, vis_str
                                vis_strt7_name = V125, vis_strt7_def = V126, vis_strt8_id = V127, vis_str
                                vis_strt9_id = V130, vis_strt9_name = V131, vis_strt9_def = V132, home_st
                                home_strt1_def = V135, home_strt2_id = V136, home_strt2_name = V137, home
                                home_strt3_name = V140, home_strt3_def = V141, home_strt4_id = V142, home
                                home_strt5_id = V145, home_strt5_name = V146, home_strt5_def = V147, home
                                home_strt6_def = V150, home_strt7_id = V151, home_strt7_name = V152, home
                                home_strt8_name = V155, home_strt8_def = V156, home_strt9_id = V157, home
                                additional_info = V160, acquisition_info = V161), envir = .GlobalEnv)
  assign(.data, dplyr::mutate(get(.data), year = substr(.data$date, start = 1, stop = 4), .before = 1),
}

##----
#PREPARE GAME LOG FUNCTION

# select and filter for only the necessary columns
# adds a column stating whether visitor or home team won
# saves time writing this as a function rather than writing the select and filter multiple times
# selects for columns in positions 1, 5, 8, and 11 through 13
# filters for extra inning games or regular inning games
# requires an existing data object
prepare_gl <- function(.data, filter_var){
  assign(.data, dplyr::select(get(.data), 1, 5, 8, 11:13), envir = .GlobalEnv)
  if(filter_var == "extra"){
    assign(.data, dplyr::filter(get(.data), length_of_game_outs > 54), envir = .GlobalEnv)
  }
  else if(filter_var == "regular"){
    assign(.data, dplyr::filter(get(.data), length_of_game_outs <= 54), envir = .GlobalEnv)
  }
  assign(.data, dplyr::mutate(get(.data), winning_team = case_when((.data$visiting_score > .data$home_s
                                                                    (.data$visiting_score < .data$home_s
         envir = .GlobalEnv)
}

##----
#PARSE RETROSHEET FUNCTION
## ------------------------------------------------------------------------
download_retrosheet <- function(season) {
  # get zip file from retrosheet website
  download.file(
    url = paste0(
      "http://www.retrosheet.org/events/", season, "eve.zip"),
    destfile = file.path(here::here("retrosheet", "zipped",
                          paste0(season, "eve.zip")))
  )
}
```

```
## ------------------------------------------------------------------------
unzip_retrosheet <- function(season) {
  # unzip retrosheet files
  unzip(file.path(here::here("retrosheet", "zipped",
                  paste0(season, "eve.zip"))),
        exdir = file.path(here::here("retrosheet", "unzipped")))
}

## ------------------------------------------------------------------------
create_csv_file <- function(season) {
  # http://chadwick.sourceforge.net/doc/cwevent.html
  # shell("cwevent -y 2000 2000TOR.EVA > 2000TOR.bev")
  wd <- getwd()
  setwd(here::here("retrosheet/unzipped"))
  cmd <- paste0("cwevent -y ", season, " -f 0-96 ",
                season, "*.EV*", " > all", season, ".csv")
  message(cmd)
  if (.Platform$OS.type == "unix") {
    system(cmd)
  } else {
    shell(cmd)
  }
  setwd(wd)
}

## ------------------------------------------------------------------------
create_csv_roster <- function(season) {
  # creates a CSV file of the rosters
  rosters <- list.files(
    path = file.path(here::here("retrosheet", "unzipped")),
    pattern = paste0(season, ".ROS"),
    full.names = TRUE)

  rosters %>%
    map_df(read_csv,
           col_names = c("PlayerID", "LastName", "FirstName",
                         "Bats", "Pitches", "Team")) %>%
    write_csv(path = file.path(here::here("retrosheet", "unzipped",
                                          paste0("roster", season, ".csv"))))
}

## ------------------------------------------------------------------------
cleanup <- function() {
  # removes retrosheet files not needed
  files <- list.files(
    path = file.path(here::here("retrosheet", "unzipped")),
    pattern = "(*.EV|*.ROS|TEAM*)",
    full.names = TRUE
  )
  unlink(files)

  zips <- list.files(
    path = file.path(here::here("retrosheet", "zipped")),
```

```
    pattern = "*.zip",
    full.names = TRUE
  )
  unlink(zips)
}

## ----eval=TRUE-----------------------------------------------------------
parse_retrosheet_pbp <- function(season) {
  download_retrosheet(season)
  unzip_retrosheet(season)
  create_csv_file(season)
  create_csv_roster(season)
  cleanup()
}
```