LINHAC EDA

David Awosoga

2023-03-06

Initial Data Setup

```
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.1.2
## Warning: package 'tibble' was built under R version 4.1.2
## Warning: package 'tidyr' was built under R version 4.1.2
## Warning: package 'readr' was built under R version 4.1.2
## Warning: package 'dplyr' was built under R version 4.1.2
## Warning: package 'stringr' was built under R version 4.1.2
```

```
library(skimr)
```

```
## Warning: package 'skimr' was built under R version 4.1.2
```

```
library(magrittr)
```

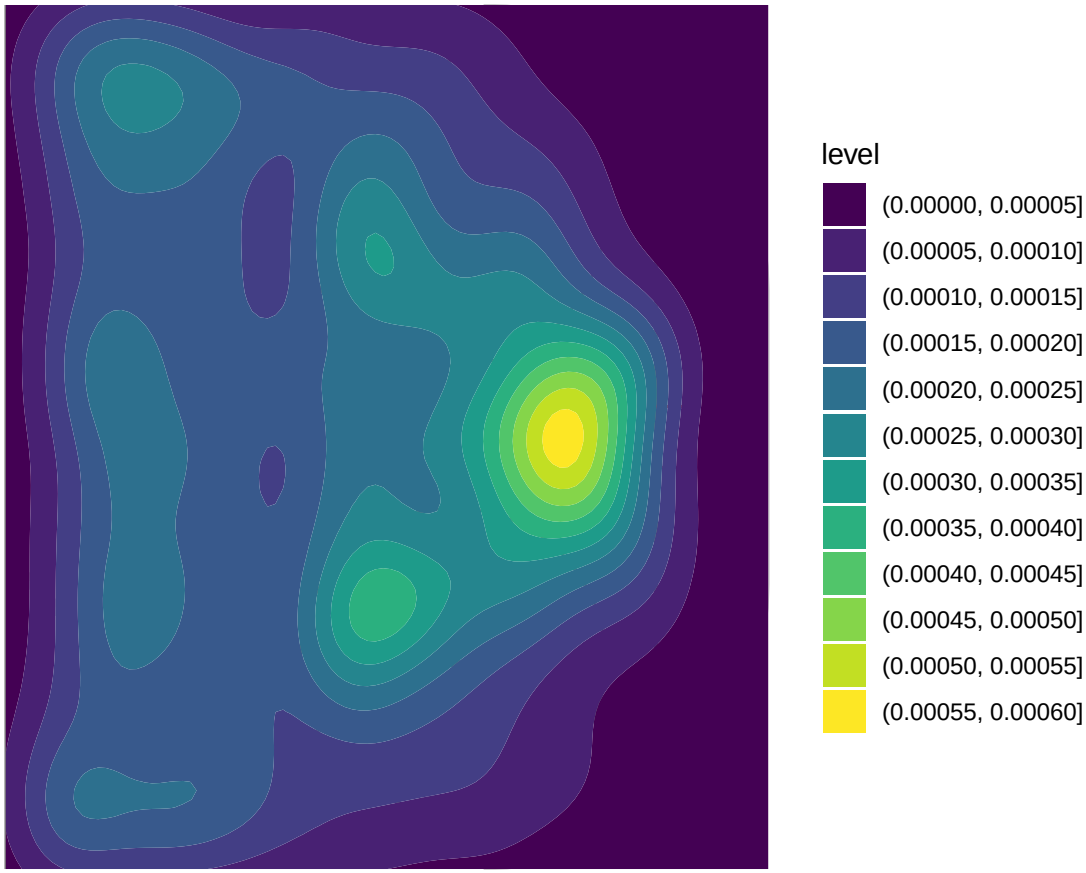
```
## Warning: package 'magrittr' was built under R version 4.1.2
```

```
df = read_csv("Linhac_df_keyed_20_games.csv", show_col_types = F)
df %<>% mutate(xadjcoord = xadjcoord+100.1126, yadjcoord = yadjcoord+42.5)
```

Here are some plots of shots

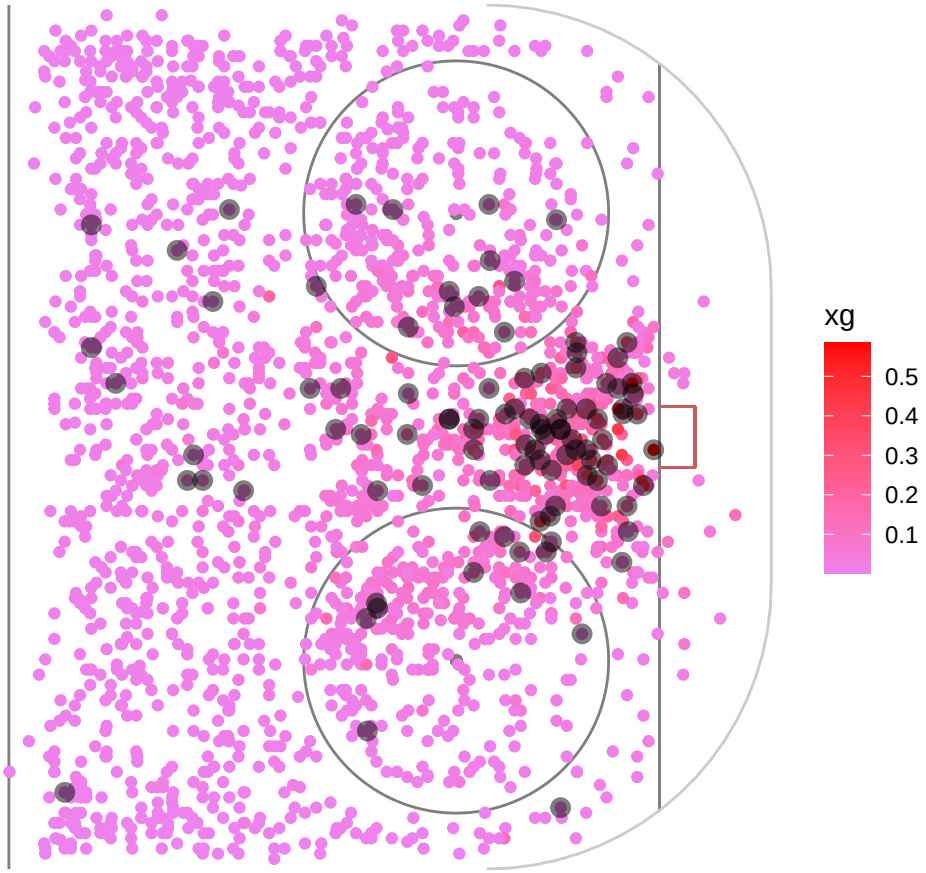
```
source("plot_rink.R")

#shot density
all_shots = df %>% filter(eventname %in% c("shot"))
plot_rink(ggplot(all_shots)) + geom_density2d_filled(aes(x = xadjcoord, y = yadjcoord)) +
  scale_x_continuous(limits = c(125, 200))
```

```
## Loading required package: ggforce
## Loading required package: cowplot
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```




```
#comparing xg to location
plot_rink(ggplot(all_shots)) + geom_point(aes(x = xadjcoord, y = yadjcoord, color = xg)) +
  geom_point(data = df %>% filter(eventname == "goal"),
             aes(x = xadjcoord, y = yadjcoord), size = 3, alpha = 0.5)+
  scale_color_gradient(high = "red", low = "violet")  +
  scale_x_continuous(limits = c(125, 200))
```

```
## Scale for x is already present.
## Adding another scale for x, which will replace the existing scale.
```



What is the shot quality like? Looking at the xg the majority aren't super great. How should I classify goals ?

```
#binning data
summary(all_shots$xg)
```

```
##     Min.  1st Qu.   Median     Mean  3rd Qu.     Max.
## 0.002164 0.008567 0.022617 0.047075 0.063966 0.583820
```

```
ranges = seq(from = 0, to = 0.6, by = 0.1)
values = all_shots %>% {table(cut(.$xg, ranges))}
values
```

```
##
##   (0,0.1] (0.1,0.2] (0.2,0.3] (0.3,0.4] (0.4,0.5] (0.5,0.6]
##      1941       218        49        10         7         5
```

```
goal_shots = (which(df$eventname == "goal") - 1)
goal_values = df[goal_shots, ] %>% {table(cut(.$xg, ranges))}
goal_values
```

```
##
##   (0,0.1] (0.1,0.2] (0.2,0.3] (0.3,0.4] (0.4,0.5] (0.5,0.6]
##        43        31        13         4         4         5
```

How should I plot goals vs xg? Is there a difference between the xg of goals and the xg of shots that don't result in a goal?

```
#maybe a boxplot?
```

How do I deal with NA's? I should probably do this systematically:

`teaminpossesion`: Possession Identifier, unique (numeric) - Corresponds to `teamid` and `opposingteamid` - `NA's` correspond to when no one has possession of the pick, perfectly correlating to a **rebound**, a **controlled entry**, a **controlled exit**, a **save**, and a **faceoff**. The outliers (or errors in the data that we might want to drop) are **assists** (2/188) and **blocks** (4/3423). I don't know what to do with **lpr** (325/15412)

Rebounds are only to do with goalies, so we can get rid of those too

```
df %>% filter(is.na(teaminpossession)) %>% nrow() #9959 initial NA
```

```
## [1] 9959
```

```
# Filter out faceoffs

df %>% filter(!(eventname %in% c("faceoff"))) %>%
  filter(is.na(teaminpossession)) %>% nrow() #Now down to 7651
```

```
## [1] 7651
```

```
df %>% filter(!(eventname %in% c("faceoff", "controlledentry",
                                 "controlledexit", "rebound" ))) %>%
  filter(is.na(teaminpossession)) %>% nrow() #Boom! Now down to 1990
```

```
## [1] 1376
```

```
#team != lead(team) & period == lead(period) ~ "possession_change"


df %>% filter(eventname == 'assist', is.na(teaminpossession))
```

```
## # A tibble: 2 x 22
##   gameid opposing~1 oppos~2 playe~3 teamg~4 teamid teami~5 curre~6    xg compi~7
##    <dbl>      <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl> <dbl>   <dbl>
## 1  88237     522672     907  724639  564869    564      NA      NA    NA   1869.
## 2  89409     288563     771  494525  505618    896      NA      NA    NA   3252.
## # ... with 12 more variables: eventname <chr>, ishomegame <dbl>,
## #   manpowersituation <chr>, opposingteamskatersonicecount <dbl>,
## #   outcome <chr>, period <dbl>, playerprimaryposition <chr>,
## #   scoredifferential <dbl>, teamskatersonicecount <dbl>, type <chr>,
## #   xadjcoord <dbl>, yadjcoord <dbl>, and abbreviated variable names
## #   1: opposingteamgoalieoniceid, 2: opposingteamid, 3: playerid,
## #   4: teamgoalieoniceid, 5: teaminpossession, 6: currentpossession, ...
```

```
df %>% filter(eventname == 'assist', !is.na(teaminpossession))
```

```
## # A tibble: 186 x 22
##    gameid opposin~1 oppos~2 playe~3 teamg~4 teamid teami~5 curre~6    xg compi~7
##     <dbl>     <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl> <dbl>   <dbl>
##  1  66445    940804     742  200689  506563    916     916      23    NA    156.
##  2  66445    940804     742  707387  506563    916     916      23    NA    159.
##  3  66445    940804     742  506563  506563    916     742     112    NA    725.
##  4  66445    940804     742  839134  506563    916     916     113    NA    728.
##  5  66445    506563     916  910446  940804    742     742     244    NA   1510.
##  6  66445    506563     916  410398  940804    742     742     244    NA   1517.
##  7  66445    940804     742  707387  506563    916     916     283    NA   1755.
```

```
##  8  66445    940804     742  242036  506563    916     916     283    NA   1758
##  9  66445    940804     742  353590  506563    916     916     556    NA   3286.
## 10  66445    940804     742  459364  506563    916     916     556    NA   3287.
## # ... with 176 more rows, 12 more variables: eventname <chr>, ishomegame <dbl>,
## #   manpowersituation <chr>, opposingteamskatersonicecount <dbl>,
## #   outcome <chr>, period <dbl>, playerprimaryposition <chr>,
## #   scoredifferential <dbl>, teamskatersonicecount <dbl>, type <chr>,
## #   xadjcoord <dbl>, yadjcoord <dbl>, and abbreviated variable names
## #   1: opposingteamgoalieoniceid, 2: opposingteamid, 3: playerid,
## #   4: teamgoalieoniceid, 5: teaminpossession, 6: currentpossession, ...
```

What is a turnover? - A change in possession that occurs after a failed pass or failed puck protection

There are two types of controlled exits - a pass and a carry.

```
df %>% select(eventname, outcome, type) %>%
  filter(eventname == "controlledexit") %>%
  table()
```

```
## , , type = carry
##
##                 outcome
## eventname        failed successful
##   controlledexit      0        304
##
## , , type = carrywithplay
##
##                 outcome
## eventname        failed successful
##   controlledexit      0       1100
##
## , , type = pass
##
##                 outcome
## eventname        failed successful
##   controlledexit   1307        236
##
## , , type = passwithplay
##
##                 outcome
## eventname        failed successful
##   controlledexit      0       1015
```

This is interesting, let's look at failed controlled exits

```
df %>% filter(eventname == "controlledexit" & outcome == "failed")
```

```
## # A tibble: 1,307 x 22
##    gameid opposin~1 oppos~2 playe~3 teamg~4 teamid teami~5 curre~6    xg compi~7
##     <dbl>     <dbl>   <dbl>   <dbl>   <dbl>  <dbl>   <dbl>   <dbl> <dbl>   <dbl>
##  1  66445    506563     916  358235  940804    742      NA      NA    NA    17.6
##  2  66445    506563     916  892235  940804    742      NA      NA    NA   266.
##  3  66445    506563     916  428581  940804    742      NA      NA    NA   325.
##  4  66445    506563     916   47709  940804    742      NA      NA    NA   334.
##  5  66445    506563     916  910446  940804    742      NA      NA    NA   341.
##  6  66445    940804     742  591556  506563    916      NA      NA    NA   346.
##  7  66445    940804     742  586302  506563    916      NA      NA    NA   352.
```

```
##  8  66445    940804     742  997285  506563    916      NA      NA    NA   412.
##  9  66445    940804     742  629919  506563    916      NA      NA    NA   449.
## 10  66445    940804     742  780695  506563    916      NA      NA    NA   458.
## # ... with 1,297 more rows, 12 more variables: eventname <chr>,
## #   ishomegame <dbl>, manpowersituation <chr>,
## #   opposingteamskatersonicecount <dbl>, outcome <chr>, period <dbl>,
## #   playerprimaryposition <chr>, scoredifferential <dbl>,
## #   teamskatersonicecount <dbl>, type <chr>, xadjcoord <dbl>, yadjcoord <dbl>,
## #   and abbreviated variable names 1: opposingteamgoalieoniceid,
## #   2: opposingteamid, 3: playerid, 4: teamgoalieoniceid, ...
```

```
#What happens next?

df[which(df$eventname == "controlledexit" & df$outcome == "failed") + 1,] %>%
  select(type) %>% table()
```

```
## .
##                  2on3              blueline              boarding
##                     1                    48                     1
##             contested                 error       errorcontested
##                    93                    94                    10
##          hipresopdump hipresopdumpcontested               hooking
##                    20                    14                     1
##                nofore                  none       northoffboards
##                    11                   242                     2
##                opdump       opdumpcontested      otherinfraction
##                   212                     5                     1
##                outlet       outletoffboards                  pass
##                    69                    40                   403
##             recovered    recoveredwithentry     recoveredwithexit
##                     4                     1                     2
##               regular               stretch      stretchoffboards
##                    19                     9                     5
```

There are lots of different types of controlled entries

```
df %>% select(eventname, outcome, type) %>%
  filter(eventname == "controlledentry") %>%
  table()
```

```
## , , type = carry
##
##                  outcome
## eventname         failed successful
##   controlledentry      0        481
##
## , , type = carrywithplay
##
##                  outcome
## eventname         failed successful
##   controlledentry      0        338
##
## , , type = carrywithplaywithshotonnet
##
##                  outcome
## eventname         failed successful
```

```
##   controlledentry      0        178
##
## , , type = carrywithplaywithshotonnetandslotshot
##
##                  outcome
## eventname         failed successful
##   controlledentry      0        205
##
## , , type = carrywithplaywithslotshot
##
##                  outcome
## eventname         failed successful
##   controlledentry      0         74
##
## , , type = carrywithshotonnet
##
##                  outcome
## eventname         failed successful
##   controlledentry      0         29
##
## , , type = carrywithshotonnetandslotshot
##
##                  outcome
## eventname         failed successful
##   controlledentry      0         39
##
## , , type = carrywithslotshot
##
##                  outcome
## eventname         failed successful
##   controlledentry      0         20
##
## , , type = pass
##
##                  outcome
## eventname         failed successful
##   controlledentry     96         84
##
## , , type = passwithplay
##
##                  outcome
## eventname         failed successful
##   controlledentry      0         53
##
## , , type = passwithplaywithshotonnet
##
##                  outcome
## eventname         failed successful
##   controlledentry      0         31
##
## , , type = passwithplaywithshotonnetandslotshot
##
##                  outcome
## eventname         failed successful
```

```
##   controlledentry      0         49
##
## , , type = passwithplaywithslotshot
##
##                  outcome
## eventname         failed successful
##   controlledentry      0         12
##
## , , type = passwithshotonnet
##
##                  outcome
## eventname         failed successful
##   controlledentry      0          5
##
## , , type = passwithshotonnetandslotshot
##
##                  outcome
## eventname         failed successful
##   controlledentry      0          3
##
## , , type = passwithslotshot
##
##                  outcome
## eventname         failed successful
##   controlledentry      0          2
```

According to the 10 rules of hockey analytics, hockey goals are pretty random corsi and fenwick are aight

My big question is looking at o-zone time

do controlled entries and exits separate these??

o-zone: all events with an x-axis > 125

```
df %>% filter(xadjcoord >= 125) %>% nrow()
```

```
## [1] 28987
```

```
#0.3812022 of the data, pretty good.

df %>% filter(xadjcoord >= 125) %>% skim()
```

Table 1: Data summary

Name	Piped data
Number of rows	28987
Number of columns	22
_____	
Column type frequency:	
character	5
numeric	17
_____	
Group variables	None

Variable type: character

skim_variable	n_missing	complete_rate	min	max	empty	n_unique	whitespace
eventname	0	1	3	16	0	22	0
manpowersituation	0	1	9	12	0	3	0
outcome	0	1	6	12	0	3	0
playerprimaryposition	0	1	1	1	0	3	0
type	0	1	4	37	0	70	0

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
gameid	0	1.00	77323.19	8649.50	60432.00	71102.00	78204.00	84953.00	89409.00	
opposingteamgoalieoniceid	28	1.00	499331.09	283807.83	4649.00	316835.00	506563.00	638522.00	996353.00	
opposingteamid	0	1.00	764.15	120.70	564.00	650.00	771.00	896.00	916.00	
playerid	0	1.00	483840.76	270465.26	6597.00	270096.00	466237.00	690783.00	997508.00	
teamgoalieoniceid	774	0.97	499059.81	289052.87	4649.00	316835.00	506563.00	638522.00	996353.00	
teamid	0	1.00	778.17	121.00	564.00	729.00	787.00	896.00	916.00	
teaminpossession	2489	0.91	777.12	121.29	564.00	729.00	787.00	896.00	916.00	
currentpossession	2489	0.91	300.01	173.18	0.00	152.00	298.00	451.00	658.00	
xg	26829	0.07	0.05	0.06	0.00	0.01	0.02	0.07	0.57	
compiledgametime	0	1.00	1834.65	1048.60	3.57	934.28	1815.10	2733.13	3900.00	
ishomegame	0	1.00	0.50	0.50	0.00	0.00	0.00	1.00	1.00	
opposingteamskatersonicecount	0	1.00	4.73	0.51	0.00	5.00	5.00	5.00	6.00	
period	0	1.00	2.03	0.83	1.00	1.00	2.00	3.00	4.00	
scoredifferential	0	1.00	-0.02	1.57	-5.00	-1.00	0.00	1.00	5.00	
teamskatersonicecount	0	1.00	4.97	0.35	0.00	5.00	5.00	5.00	6.00	
xadjcoord	0	1.00	159.70	23.73	125.02	136.52	160.66	179.88	200.00	
yadjcoord	0	1.00	42.44	28.62	0.00	15.09	41.75	70.42	85.00	

due to the standardized nature of the data, I can assume that these are just the team on offense

When to shoot? I gotta figure out how to segment/group by possessions. But first, I need to fill out the na's.

YOOOOOOO I just realized that the na's for team in possession happen at events where there are multiple pieces of information to describe the same timestamp!! The controlled entries and exits provide key bits of contextual information, but aren't new events.

Still gotta check for duplicates via lpr's.

```
df %>% filter(xadjcoord >= 124, !(eventname %in% c('faceoff', 'controlledexit'))) %>% filter(is.na(team
  select(eventname) %>% table()
```

```
## .
##          assist           block controlledentry             lpr          rebound
##               2               1            1604             123               2
##            save
##              14
```

What should I do with these? 1. I think that I can filter out faceoffs since they aren't continuations of possessions 2. I can filter out controlled exits since they are

What events result in an end of possession? - failed puck protection

Do assists have the same coordinates with passes?

How many distinct offensive zone possessions are there? First, how many controlled entries are there? Maybe I should focus on those? Other ways to have the puck on offense is a dump in and a turnover or winning an lpr

```
df %>% filter(eventname == "controlledentry") %>% select(xadjcoord) %>% summary()
```

```
##    xadjcoord
##  Min.   : 21.35
##  1st Qu.:125.46
##  Median :125.56
##  Mean   :125.14
##  3rd Qu.:126.07
##  Max.   :168.31
```

Okay so what if we look at possesions following a controlled entry? How many are there? There are 1699 controlled entries and 1609 controlledentryagainst. Which events do these happens

```
entries = which(df$eventname == "controlledentry")
df %>% select(eventname) %>% table()
```

```
## .
##                 assist                  block                  carry
##                    188                   3423                   4375
##                  check        controlledentry controlledentryagainst
##                   1120                   1699                   1609
##         controlledexit                 dumpin                dumpout
##                   3962                   1173                   1382
##                faceoff                   goal                  icing
##                   2308                    106                    167
##                    lpr                offside                   pass
##                  15412                     81                  17455
##                penalty           penaltydrawn         puckprotection
##                    135                    131                   3971
##                rebound             reception                   save
##                    614                  13414                   1045
##                   shot                  sogoal       sopuckprotection
##                   2230                      6                      3
##                 soshot
##                     32
```

The anonymization of data would lead us towards focusing on the team rather than individual players

takeaways: steals, pass interceptions, and won puck battles that result in a change of possession

we could make an xg model at any point on the ice using the filled out values as our test and train set?

definitng defensive actions as takeaways or puck recoveries in the opposing team's offensive zone. Every other action is offensive

xg of shots taken as a function of time of possession. Expect quadratic?

Look at passing?

```
#Are there failed receptions?

df %>% filter(eventname == "pass") %>% select(outcome) %>% table()
```

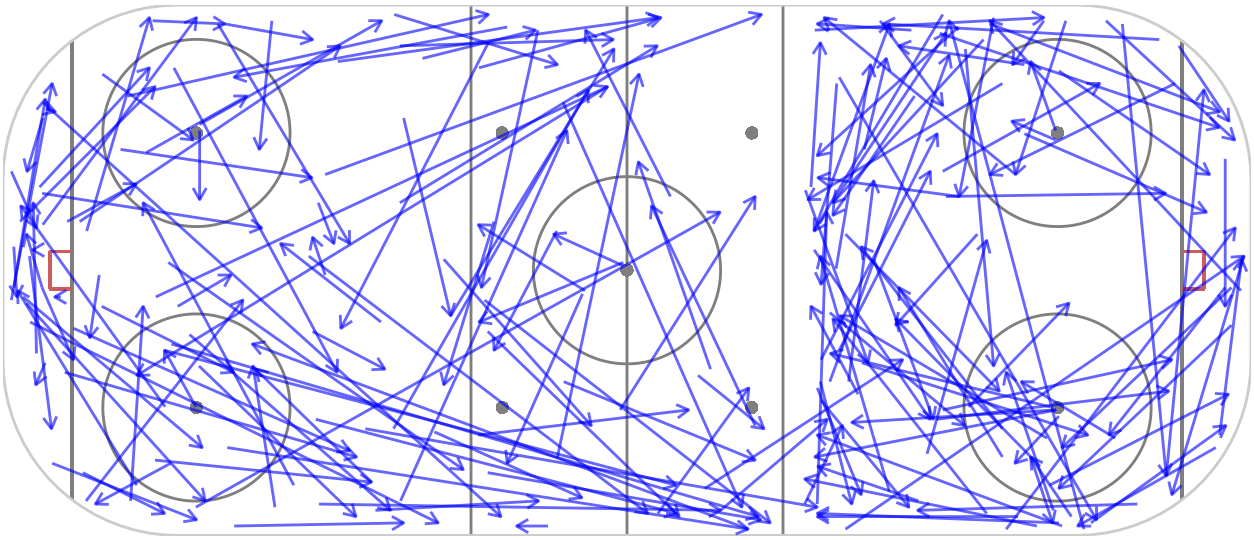
```
## .
##       failed   successful undetermined
##         4465        12954           36
```

```
df %>% filter(eventname == "reception") %>% select(outcome) %>% table()
```

```
## .
##     failed successful
##        460      12954
```

So, successful passes are paired with successful receptions. Let's plot those

```
#assumption is that passes and receptions are consecutive
p = df %>% filter(eventname %in% c("pass") & outcome == "successful") %>%
  transmute(x_1 = xadjcoord, y_1 = yadjcoord)

r = df %>% filter(eventname %in% c("reception") & outcome == "successful") %>%
  transmute(x_2 = xadjcoord, y_2 = yadjcoord)

d = cbind(p,r)

#Find Pass distances
#d %>% mutate(pass_distance = sqrt((x_1 - x_2)^2 + (y_1 - y_2)^2))

#plot shots
plot_rink(ggplot(head(d, n = 200))) +
  geom_segment(aes(x = x_1, xend = x_2, y = y_1, yend = y_2),  color = "blue", alpha = 0.6, arrow = arr
```



```
#There are 2 types of receptions: ozentry and regular

df %>% filter(eventname %in% c("reception") & outcome == "successful")  %>%
  select(type) %>% table()
```

```
## .
## ozentry regular
##     250   12704
```

```
#There are 19 types of (successful) passes

df %>% filter(eventname %in% c("pass") & outcome == "successful")  %>%
  select(type) %>% table()
```

```
## .
```

```
##                     d2d            d2doffboards                 eastwest
##                    1891                     798                     934
##       eastwestoffboards                   north          northoffboards
##                      19                    1411                     914
##                  outlet         outletoffboards                 ozentry
##                    2084                     525                     177
##        ozentryoffboards          ozentrystretch ozentrystretchoffboards
##                      31                      26                      16
##                    rush           rushoffboards                    slot
##                     230                      16                     611
##                   south          southoffboards                 stretch
##                    2365                     429                     380
##        stretchoffboards
##                      97
```

What about failed passes?

```
failed_passes = which(df$eventname %in% c("pass") & df$outcome == "failed")

#df %>% lead(failed_passes, n = 2
```

How many unique possessions are there?

```
possessions = numeric()
for(i in unique(df$gameid)) {
  possessions = append(possessions, df %>% filter(gameid == i) %>% select(currentpossession) %>% unique
}

sum(possessions) #there are about 11895 (including NA's, so maybe subtract 20)
```

```
## [1] 11895
```

What types of goals were scored?

```
df %>% filter(eventname == "goal") %>% select(manpowersituation) %>% table()
```

```
## .
## evenStrength    powerPlay  shortHanded
##           70           32            4
```

When to pass or carry? We don't have contextual data of who's pressuring you but maybe we can proxy/ignore it?

LEAD AND LAG HAVE BEEN WHAT I'VE BEEN LOOKING FOR INSTEAD OF THE WACKY INDEXING?

NEST AND UNNEST COULD ALSO BE COOL

#Literature Review

After doing some reading I guess that the Markov Decision Process will be too hard The hidden markov model doesn't seem to have enough areas for new application

All these people are building an xg model from scratch. We have one given to us, so we should probably use it

A lot of these things use the previous two events to predict whether a goal will be scored or conceded in one of the next 10 events. Maybe we could extend it?

I need to do a lead on controlled entries and exits