关于数据：

该数据集涵盖了科比布莱恩特20年职业生涯的所有投篮数据，其中，有5000个投篮的命中情况（是否命中）在题目中是缺失项。我们的目标就是利用剩下的投篮数据来预测这5000个投篮是否命中。

我们利用R来进行数据的预处理。

数据预处理：

1. 首先我们将需要用到的R程序包载入，并将所给的数据读入R程序。

In [1]:

#载入data.table、lubridate、dummies包

library(data.table)

library(lubridate)

library(dummies)

library(dplyr)

library(ggplot2)

In [2]:

#读取数据

data1 = fread("data.csv")

In [3]:

#列出变量

names(data1)

Out [3]:

[1] "action\_type" "combined\_shot\_type" "game\_event\_id"

[4] "game\_id" "lat" "loc\_x"

[7] "loc\_y" "lon" "minutes\_remaining"

[10] "period" "playoffs" "season"

[13] "seconds\_remaining" "shot\_distance" "shot\_made\_flag"

[16] "shot\_type" "shot\_zone\_area" "shot\_zone\_basic"

[19] "shot\_zone\_range" "team\_id" "team\_name"

[22] "game\_date" "matchup" "opponent"

[25] "shot\_id"

我们先来简要解释一下这些变量的含义。

* action\_type：投篮动作的细分。是字符型变量。
* combined\_shot\_type：投篮动作的大分类。例如跳投、扣篮、上篮等。是字符型变量。
* game\_event\_id：某一场比赛内发生事件的ID。是数值型变量。
* game\_id：每场比赛的ID。是数值型变量。
* lat：纬度，和球员所处在球场的位置有关。是数值型变量。
* loc\_x：球员位置的x坐标。是数值型变量。
* loc\_y：球员位置的y坐标，是数值型变量。
* lon：经度。和球员所处在球场的位置有关。是数值型变量。
* minutes\_remaining：投篮发生时该节剩余的分钟数。是数值型变量。
* period：投篮发生时在第几节。是数值型变量。
* playoffs：该比赛是否是季后赛。是数值型变量。
* season：该比赛所处的赛季。是日期型变量。
* seconds\_remaining：投篮发生时该节剩余的秒数。是数值型变量。
* shot\_distance：投篮距离篮筐的距离。是数值型变量。
* shot\_made\_flag (this is what you are predicting)：投篮是否命中，1为命中，0为不中。这是最终要预测的变量。
* shot\_type：投篮类型（是2分还是3分）。是字符型变量。
* shot\_zone\_area：投篮的区域（是在球场左边、中间还是右边等）。是字符型变量。
* shot\_zone\_basic：投篮的区域（是远投、中距离、还是在禁区等）。是字符型变量。
* shot\_zone\_range：投篮的距离（是在多少尺范围内）。是字符型变量。
* team\_id：该球员所在的球队ID。是数值型变量。
* team\_name：该球员所在的球队名。是字符型变量。
* game\_date：比赛日期。是日期型变量。
* matchup：对阵双方。是字符型变量。
* opponent：所对阵的球队的队名。是字符型变量。
* shot\_id：投篮的ID。是数值型变量。

我们先作图来直观地观察这些数据体现的内容。

编写courtplot函数来描绘出投篮的位置。

先以combined\_shot\_type这一项来观察：

In [4]:

train <- data1[!is.na(data1$shot\_made\_flag),]

train$shot\_made\_flag = as.factor(train$shot\_made\_flag)

courtplot = function(feat) {

feat <- substitute(feat)

train %>%

ggplot(aes(x = lon, y = lat)) +

geom\_point(aes\_q(color = feat), alpha = 0.7, size = 3) +

ylim(c(33.7, 34.0883)) +

scale\_color\_brewer(palette = "Set1") +

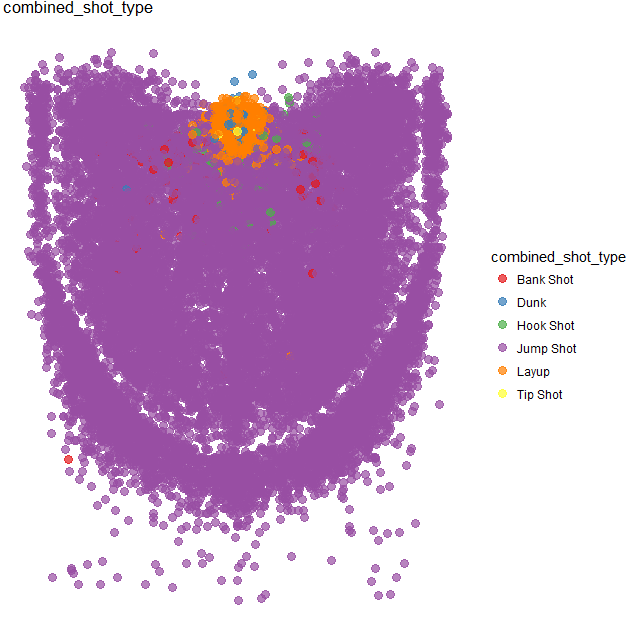
theme\_void() +

ggtitle(paste(feat))

}

courtplot(combined\_shot\_type)

Out [5]:



由于jump shot的数量过多，比较难看出其他投篮类型的分布情况。于是我们将jump shot的点隐去进一步观察：

In [6]:

ggplot() +

geom\_point(data = filter(train, combined\_shot\_type == "Jump Shot"),

aes(x = lon, y = lat), color = "grey", alpha = 0.3, size = 2) +

geom\_point(data = filter(train, combined\_shot\_type != "Jump Shot"),

aes(x = lon, y = lat,

color = combined\_shot\_type), alpha = 0.7, size = 3) +

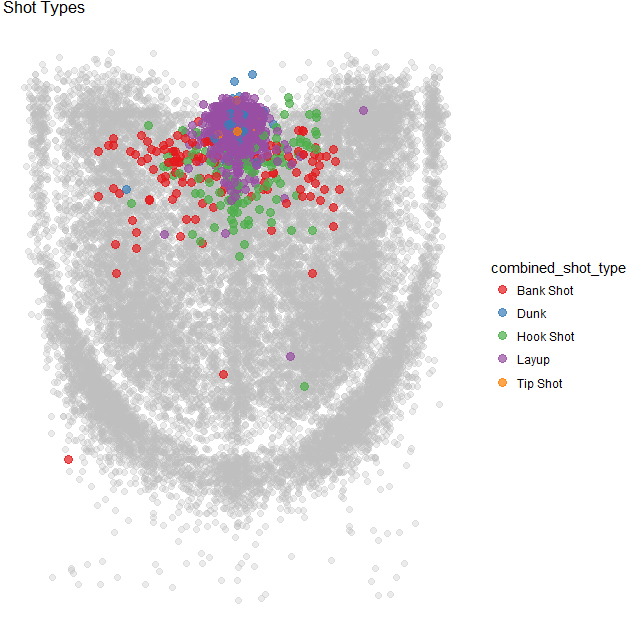
ylim(c(33.7, 34.0883)) +

scale\_color\_brewer(palette = "Set1") +

theme\_void() +

ggtitle("Shot Types")

Out [6]:

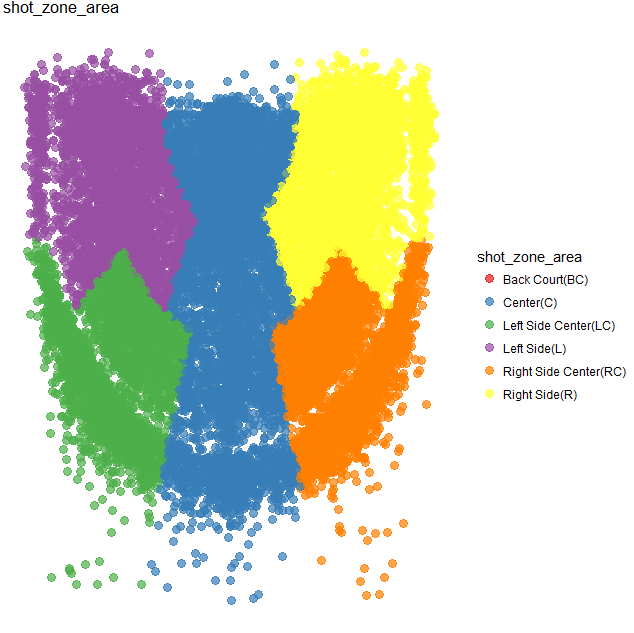


再以shot\_zone\_area 这一项来观察：

In [7]:

courtplot(shot\_zone\_area)

Out [7]:

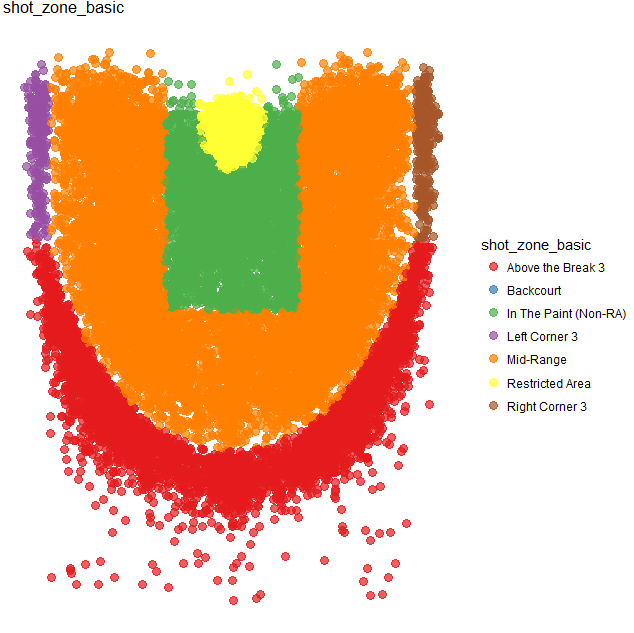


以shot\_zone\_basic这一项来观察：

In [8]:

courtplot(shot\_zone\_basic)

Out [8]:

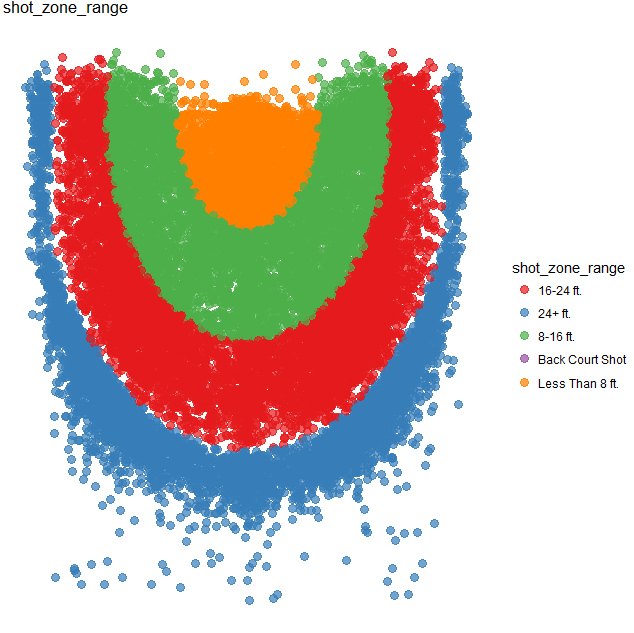


以shot\_zone\_range这一项来观察：

In [9]:

courtplot(shot\_zone\_range)

Out [9]:



这样，我们就对这组投篮数据有了直观的理解。

我们可以大致将这些数据分成两组。

数值型变量：lat, loc\_x, loc\_y, lon, minutes\_remaining, seconds\_remaining, shot\_distance

字符型变量：action\_type, combined\_shot\_type, period, playoffs, season, shot\_type, shot\_zone\_area, shot\_zone\_basic, shot\_zone\_range

对于日期型，我们需要单独进行处理。

1. 为了能够确定每次投篮的位置，便于训练。我们将投篮的位置坐标进行z-score标准化。

经过z-score标准化后的数据符合标准正态分布。其转化函数为：

其中μ为所有样本数据的均值，σ为所有样本数据的标准差。

In [10]:

#对投篮的位置坐标进行标准化

data1[,lon:=with(data1 , (data1$lon-mean(data1$lon))/sd(data1$lon))]

data1[,lat:=with(data1 , (data1$lat-mean(data1$lat))/sd(data1$lat))]

data1[,loc\_x:=with(data1 , (data1$loc\_x-mean(data1$loc\_x))/sd(data1$loc\_x))]

data1[,loc\_y:=with(data1 , (data1$loc\_y-mean(data1$loc\_y))/sd(data1$loc\_y))]

1. 为了使各个数值型变量在同一个标准下进行训练。我们对其进行去量纲化处理。方法是是使用min-max标准化。其是对原始数据的线性变换，使结果落到[0,1]区间，转换函数如下：

其中max为样本数据的最大值，min为样本数据的最小值。

In [11]

#对数值型的数据进行min-max标准化

data1[,time:=with(data1 , (data1$minutes\_remaining/60)+data1$seconds\_remaining)]

data1[,time:=with(data1,((data1$time-min(data1$time)))/(max(data1$time)-min(data1$time)))]

data1[,shot\_distance:=with(data1,((data1$shot\_distance-min(data1$shot\_distance)))/(max(data1$shot\_distance)-min(data1$shot\_distance)))]

data1[,minutes\_remaining:=NULL]

data1[,seconds\_remaining:=NULL]

data1[,game\_event\_id:=with(data1,((data1$game\_event\_id-min(data1$game\_event\_id)))/(max(data1$game\_event\_id)-min(data1$game\_event\_id)))]

data1[,game\_id:=with(data1,((data1$game\_id-min(data1$game\_id)))/(max(data1$game\_id)-min(data1$game\_id)))]

data1$game\_id= NULL

data1$game\_event\_id= NULL

data1[,period:=with(data1,((data1$period-min(data1$period)))/(max(data1$period)-min(data1$period)))]

data1[,season:=NULL]

data1[,shot\_id:=NULL]

1. 对于日期型的变量，我们利用R程序内的lubridate包内的lubridate函数将game\_date内的年、月、日化为数值型，再进行max-min标准化。

In [12]:

#将日期变量化为数值型，再进行max-min标准化

data1[,month:=lubridate::month(game\_date)]

data1[,year:=lubridate::year(game\_date)]

data1[,wday:=lubridate::wday(game\_date)]

data1[,game\_date:=NULL]

data1[,wday:=with(data1,((data1$wday-min(data1$wday)))/(max(data1$wday)-min(data1$wday)))]

data1[,year:=with(data1,((data1$year-min(data1$year)))/(max(data1$year)-min(data1$year)))]

data1[,month:=with(data1,((data1$month-min(data1$month)))/(max(data1$month)-min(data1$month)))]

1. 由于一些训练方法（例如xgboost）仅适用于数值型变量，所以对于那些非数值型的变量，需要变成哑变量。我们利用R程序内的dummies包将那些字符型变量全部转化成哑变量。其中，matchup那一列数据需要先将其中反映主客场的信息提取出来，再进行哑变量处理。

In [13]:

#将非数值型的变量化为哑变量

data1[matchup %like% "@", matchup := 'Away']

data1[matchup %like% "vs.", matchup := 'Home']

data1$playoffs= as.character(data1$playoffs)

df = dummy.data.frame(data1 , names = c('action\_type' , 'combined\_shot\_type' , 'shot\_type' , 'shot\_zone\_area' ,'playoffs' , 'shot\_zone\_basic' , 'shot\_zone\_range' , 'matchup' , 'opponent') , sep='\_')

至此，我们已经完成了数据预处理。