An introduction to R

Scott Powers and Ryan Wang

January 7, 2014

Goal

- Serious data work requires programming (at least with the current state of tools like Excel and Tableau)
- In this tutorial, we'll work through the basics of R by analyzing the set of box scores from the '12-'13 NBA season
- Among other things, we will:
  1. Identify the top scorers in that season
  2. Understand how statistics vary over the course of a season
  3. Determine the relationship between scoring and rebounding
  4. Build a model for predicting this year's top scorers

The help function

- R is a very well-documented language
- A crucial function is help, as it allows you to access that documentation
- It can be accessed by help(function) or ?function for short e.g. type ?help to learn more about the help function
- Google is also a great resource and usually links to either stackoverflow or the R mailing list

Outline

Data frames

Let's start by reading in some real-life data

```
>   setwd('~/GitHub/data-mining-intersession/day1')
>   nba <- read.csv('box_2012to2013.csv', stringsAsFactors=
>   class(nba)
```

```
[1] "data.frame"
```

In R terminology, data typically comes in the form of a *data.frame* (try ?data.frame). Data frames are essentially matrices that can store non-numeric values, and very similar to spreadsheets.

Data frames cont.

This is essentially the same as what you would see if you opened the .csv file in MS Excel

```
>   nba[1:5,1:5]
```

```
    game_id  pid         player pos                  team
1 400277721 4270 Trevor Booker  PF Washington Wizards
2 400277721 2426  Trevor Ariza  SF Washington Wizards
3 400277721 2399  Emeka Okafor   C Washington Wizards
4 400277721 4010    A.J. Price  PG Washington Wizards
5 400277721 6580  Bradley Beal  SG Washington Wizards
```

Columns correspond to *variables* e.g. column 5 indicates the number of minutes played. Rows correspond to *observations* e.g. row 1 corresponds to Trevor Booker's line on the box score.

Data frames cont.

This data frame contains 31,375 rows (observations) and 19 columns (variables)

```
> nrow(nba)
[1] 31351
> ncol(nba)
[1] 19
> dim(nba)
[1] 31351    19
```

Outline

Accessing elements of the data

Specific elements of the data can be accessed with [row, column] coordinates

```
>   nba[1:2,1:2]

    game_id  pid
1 400277721 4270
2 400277721 2426

>   nba[1,1]

[1] 400277721

>   nba[2,1]

[1] 400277721
```

We'll discuss subsetting data frames more in-depth after we have seen how vectors (and specifically sequences) work.

Vectors

The fundamental unit of computation in R is the vector. Create a vector by combining values with the "c" function (see ?vector and ?c):

```
>   c(1,2,3,4,5)
[1] 1 2 3 4 5
>   c('a','b','c','d','e')
[1] "a" "b" "c" "d" "e"
```

Vectors cont.

Assign a value, which includes vectors, to a name using the "<-" operator (see ?assignOps)

```
>   v1 <- c(1,2,3,4,5)
>   v1

[1] 1 2 3 4 5

>   v1[1]

[1] 1
```

The first element of v1 is 1, and the second element is 2. Note that unlike most programming languages, indexing starts at 1.

Vectors cont.

Most functions involving vectors operate element-wise. Calling a function like ?sum or ?mean on a vector will collapse it:

```
>   v1 + 3
[1] 4 5 6 7 8
>   v1 + v1
[1]  2  4  6  8 10
>   sqrt(v1)
[1] 1.000000 1.414214 1.732051 2.000000 2.236068
>   sum(v1^2)
[1] 55
```

Vectors cont.

The ":" operator (see ?colon) creates a sequence, which is also a vector

```
>   x <- 1:5
>   x

[1] 1 2 3 4 5

>   x[1]

[1] 1
```

Vectors cont.

Boolean values are TRUE and FALSE, or T and F for short (see ?logical)

```
> TRUE & FALSE
```

```
[1] FALSE
```

```
> !(!T & F)
```

```
[1] TRUE
```

It is easy to create boolean vectors by writing conditions. This will come in handy soon.

```
> 1:5 > 2
```

```
[1] FALSE FALSE  TRUE  TRUE  TRUE
```

Exercises

1. Create a vector of the even numbers between 1 and 50
2. What happens if you use ?sum on a vector of boolean values?
3. What would happen if you typed nba[,]?

Outline

Accessing elements of the data cont.

Now that we know about sequences, see that the data frame is subsetted by passing a vector of row and/or column indices

```
>   nba[1:5,1:5] # rows 1 to 5, columns 1 to 5
```

```
    game_id  pid         player pos                team
1 400277721 4270 Trevor Booker  PF Washington Wizards
2 400277721 2426  Trevor Ariza  SF Washington Wizards
3 400277721 2399  Emeka Okafor   C Washington Wizards
4 400277721 4010    A.J. Price  PG Washington Wizards
5 400277721 6580  Bradley Beal  SG Washington Wizards
```

```
>   nba[1:5,] # would give rows 1 to 5, all columns
```

You can access all the entries along a particular dimension by leaving the slot blank.

Accessing elements of the data cont.

A neat feature of data frames is that columns are named.
Access specific columns with the "$" operator (see ?Extract)

```
>   names(nba)[1:5]
[1] "game_id" "pid"     "player"  "pos"     "team"
>   nba$pid[1:5]
[1] 4270 2426 2399 4010 6580
>   nba[1:5, 'pid'] # alternatively...
[1] 4270 2426 2399 4010 6580
```

Exploring the data

Using boolean vectors, we can subset the data to look at specific players' statistics

```
>   lbj <- nba[nba$player == "LeBron James",]
>   basic <- c('player','min','pts','reb')
>   lbj[1:5,basic]
```

```
            player min pts reb
40    LeBron James  29  26  10
464   LeBron James  37  23   7
735   LeBron James  39  20   9
1097  LeBron James  30  23  11
1406  LeBron James  30  20  12
```

Under the hood, the boolean vector is being translated into a vector of indices for the TRUE entries. This can be accomplished explicitly by using the ?which function on a boolean vector.

Exploring the data cont.

It's not enough to know only the dimensions of the data frame, which tell us the number of box scores. We might need to know things such as: how many players were active?

```
>   v1 <- c('a','a','b','d')
>   unique(v1)

[1] "a" "b" "d"

>   length(unique(v1))

[1] 3

>   length(unique(nba$pid)) # number of players active

[1] 476
```

The ?unique function returns the unique values that appear in the inputted vector. The ?length function returns its length.

Exercises!

1. Familiarize yourself with the nba dataset. What does it contain? (Try ?View to for a spreadsheet-like view)
2. Create a subset of the data that contains entries for all point guards (PG). What are its dimensions?
3. How many total games were played in the '12-'13 season? (note: some questions don't require data)
4. What is the average number of minutes played? Points scored? (Try ?mean)

Outline

Total points scored

We now have some basic tools for exploring and manipulating data. We can use them to answer question such as: how many points did LeBron James score in the '12-'13 season?

```
>   sum(lbj$pts)
[1] 2036
>   sum(nba$pts[nba$player=='LeBron James'])
[1] 2036
```

Check this against ESPN's records `http://espn.go.com/nba/player/stats/_/id/1966/lebron-james`.

Points per game (PPG)

Let's try answering more substantive questions such as: Who had the highest PPG? First we need to remove games in which a player did not play.

```
>   nba <- nba[nba$min>0,]
```

Next, we want to repeatedly do the following:

```
>   pid <- 9
>   tmp <- nba[nba$pid==pid,]
>   mean(tmp$pts)
```

```
[1] 10.92405
```

Player 9 is Ray Allen. How would you check this?

Points per game (PPG) cont.

We could literally repeat the averaging process with a loop. But since this is so common in data work, R has fast functions for this type of thing.

```
>   ppg <- aggregate(x=nba$pts,
+                    by=list(pid=nba$pid),
+                    FUN=mean)
>   max(ppg$x) # highest ppg is 28.6
```

```
[1] 28.65672
```

What's going on here? The vector nba$pts is first split into subsets according to nba$pid. Then the ?mean function is being applied to each subset. This idea is known as split-apply-combine and is similar to map-reduce (which is a big buzzword in big data).

Points per game (PPG) cont.

We can identify this player by figuring out the index of the maximum value, and looking it up in nba.

```
>   idx <- which.max(ppg$x)
>   pid <- ppg$pid[idx]
>   pid
[1] 1975
>   nba$player[nba$pid==pid][1]
[1] "Carmelo Anthony"
```

Points per game (PPG) cont.

What if we want to know the top 5 scorers? The top N scorers? Luckily there is a better way than manually looking everyone up... First, we'll create a new dataset that maps pid to player name.

```
>   plyrs <- unique(nba[,c('pid','player')])
>   plyrs[1:5,]
```

```
   pid         player
1 4270 Trevor Booker
2 2426  Trevor Ariza
3 2399  Emeka Okafor
4 4010    A.J. Price
5 6580  Bradley Beal
```

This is a lookup table of sorts. We can always consult it to figure out the player name corresponding to a pid. Can we always do the reverse?

Points per game (PPG) cont.

Next, let's try merging two datasets. We'll combine our ppg dataset with the pid->player mapping so we don't have to lookup names separately anymore.

```
>   ppg <- merge(ppg, plyrs, by='pid')
>   ppg[1:5,]
```

```
  pid          x              player
1   9 10.924051           Ray Allen
2  25 12.386667 Metta World Peace
3  63  8.363636  Chauncey Billups
4  91  7.194444        Elton Brand
5 110 27.346154        Kobe Bryant
```

Think about how you would do this by hand. For each pid in *ppg*, you'd look it up in *plyrs* and find the associated player name. Then you'd come back to *ppg* and record the name you found. This is also known as joining.

Points per game (PPG) cont.

The last thing to do is sort ppg. We'll use the ?order function to sort by ppg$x.

```
>   order_ppg <- order(ppg$x, decreasing=T)
>   order_ppg[1:5] # vector of indices

[1]  79 201   5  78 278

>   ppg <- ppg[order_ppg,]
>   ppg[1:5,]

     pid        x           player
79  1975 28.65672 Carmelo Anthony
201 3202 28.14815    Kevin Durant
5    110 27.34615     Kobe Bryant
78  1966 26.78947    LeBron James
278 3992 25.93590    James Harden
```

Exercises

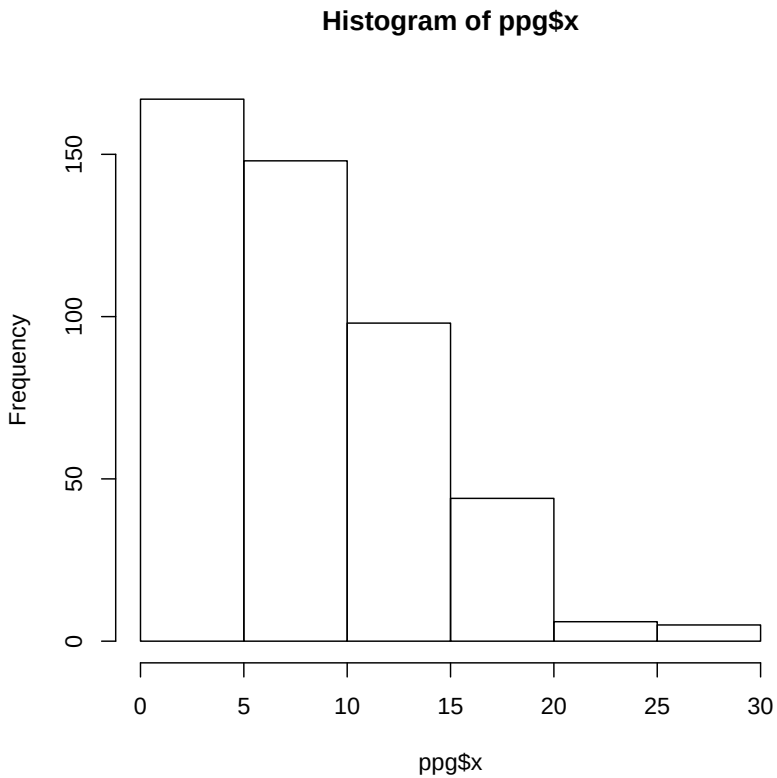
1. Calculate rebounds, assists, and blocks per game (reb, ast, blk). Who are the top players in each category?
2. Which team scored the most points per game?

Outline

Histograms

We know a bit about PPG for the top players. What does the distribution look like?

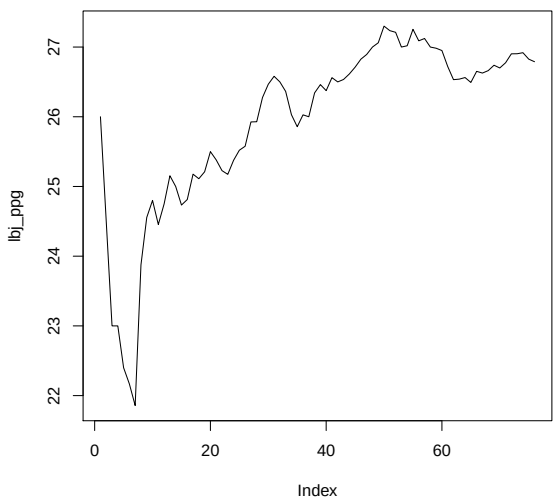
```
> hist(ppg$x)
```



PPG over the season

Another interesting question is how much does PPG vary over the course of the season? We'll plot the time series with ?plot.

```
>   lbj <- subset(nba, player=='LeBron James')
>   lbj_ppg <- cumsum(lbj$pts)/1:length(lbj$pts)
>   plot(lbj_ppg, type='l')
```

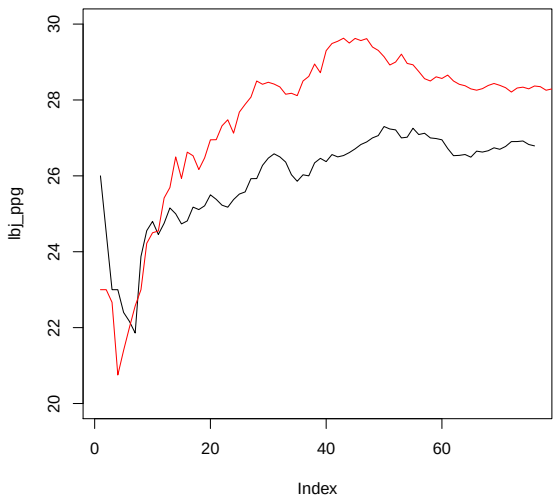


Time series are very common in applications within finance.

PPG over the season cont.

How did LeBron compare to Kevin Durant? We can overlay plots with the ?lines function

```
>   kd <- subset(nba, player=='Kevin Durant')
>   kd_ppg <- cumsum(kd$pts)/1:length(kd$pts)
>   plot(lbj_ppg, type='l', ylim=c(20,30))
>   lines(kd_ppg, type='l', col='red')
```



Note that we're sidestepping the fact that they might not have played the same number of games.

Relationship between scoring and rebounding

Let's switch gears a bit. We'll now consider the relationship between scoring and rebounding. First let's get all these stats in one place.

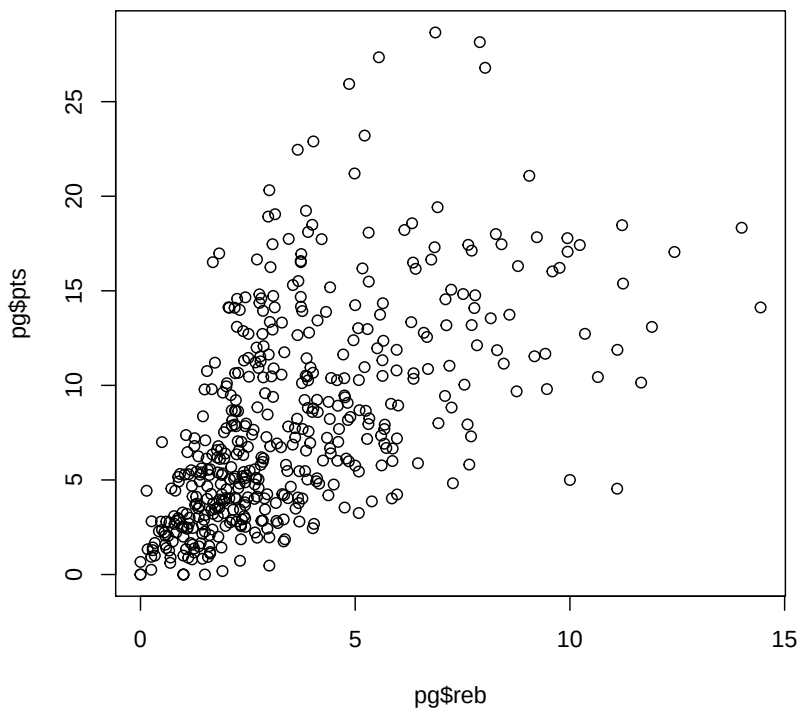
```
>   stats <- c('min','reb','ast','stl','blk','tos','pts')
>   pg <- aggregate(nba[,stats],list(pid=nba$pid),mean)
>   pg <- merge(pg, plyrs, by='pid')
>   pg[1:5,1:5]
```

```
  pid      min      reb      ast      stl
1   9 25.73418 2.746835 1.7088608 0.8481013
2  25 33.74667 4.960000 1.4933333 1.6266667
3  63 19.00000 1.454545 2.2272727 0.5454545
4  91 21.25000 5.972222 0.9722222 0.6944444
5 110 38.69231 5.551282 6.0128205 1.3589744
```

Relationship between scoring and rebounding cont.

Next, let's look at a scatter plot of scoring (Y axis) versus rebounding (X axis). There seems to be a positive relationship

```
>   plot(y=pg$pts, x=pg$reb)
```



Exercises

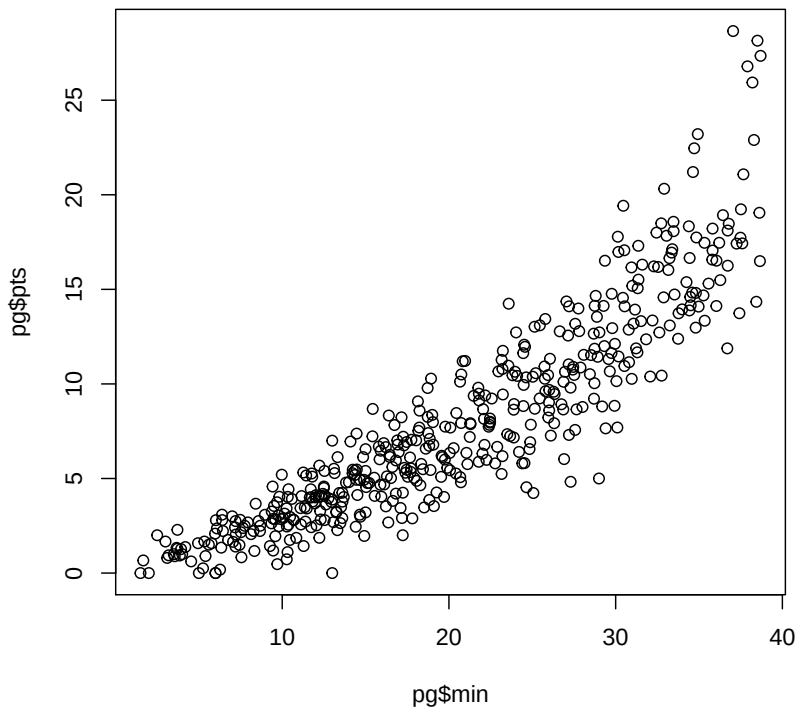
1. Plot a histogram of rebounds per game. What does it tell you?
2. How else can you describe distributions? (Try ?mean, ?sd, ?quantile).
3. Plot the relationship between various stats. What is a drawback of per game stats? (Hint: Try against minutes played)

Outline

Scoring and minutes

The point of Exercise 3 was that all of the per game statistics are highly dependent (and nearly linearly) on minutes played.

```
> plot(pg$pts~pg$min)
```



Scoring and minutes cont.

We can quantify the relationship between scoring and minutes using linear regression, implemented with the ?lm function (for linear model).

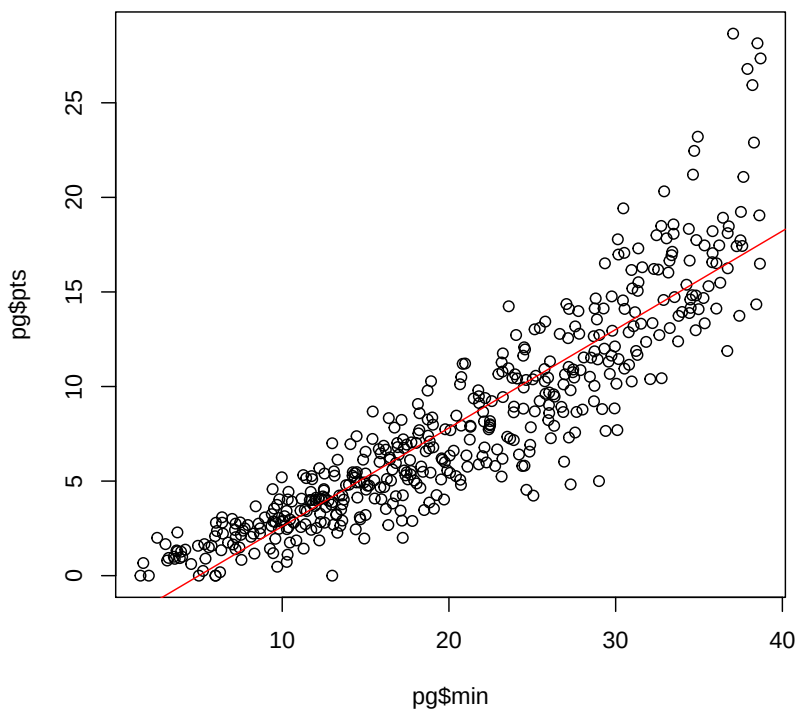
```
>   m1 <- with(pg, lm(pts~min))
>   m1

Call:
lm(formula = pts ~ min)

Coefficients:
(Intercept)          min
    -2.6043       0.5206
```

Try summary(m1) for an assessment of how well the model fits the data. This is the type of thing you would learn about in an intro stats course.

Scoring and minutes cont.

The idea of linear regression is basically to fit a line through the scatter plot of Y on X.

```
>   plot(pg$pts~pg$min)
>   abline(m1, col='red')
```



Scoring and minutes cont.

How do we interpret this model?

```
>   m1

Call:
lm(formula = pts ~ min)

Coefficients:
(Intercept)          min
    -2.6043       0.5206
```

Since it's a line, the model is defined by two parameters - slope and intercept. The slope can be interpreted as follows: for the average player, each additional minute per game results in an additional 0.5 points per game.

Scoring and minutes cont.

Linear regression supplies us with a method for making predictions.

```
>   fake_player <- list(min=20.0)
>   predict(m1, newdata=fake_player)
       1
7.807913
```

In particular, $\hat{pts} = 0.52 * min - 2.60$ where $\hat{pts}$ is our prediction. So we would predict that a player who played 20 minutes per game would score 7.8 points per game. Admittedly, this is not a great model. But it's a start.

Exercises

1. Extend the regression model by adding additional variables. (Hint: You will need formulas of the form lm(y x1+x2+...))
2. (challenge) Rank top scorers by points per 48 minute, as opposed to points per game. What additional considerations must you make?
3. (challenge) Compare points per 48 minute to rebounds per 48 minute. Verify that the positive relationship between ppg and rpg is due almost entirely to minutes played.
4. (challenge) Create a regression model for predicting player's ppg from last year's ppg. Data from the '11-'12 season is also available on GitHub. How well does this model work for '13-'14?