Kaggle Kobe Bryant Analysis

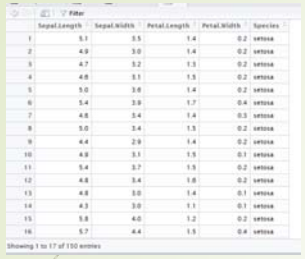
ACM Meetup

Norm Zeck

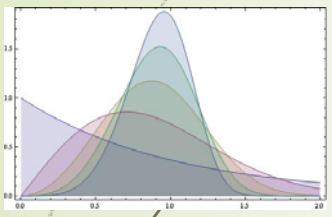
My goals with the project & this talk

- Project: Experimental exercise
  - Modeling, Feature Selection, Caret package, XGBoost
  - Data set that would stress predictive algorithms
  - Choose a set that I had domain knowledge
- Talk: Walk through of a sample data science project
  - Background and terminology
  - Development steps
  - Value of knowing the domain
  - Exploratory Stage: Use of visualization & analysis
  - Modeling Stage: results, assessment, sample details
  - Caveat on Kaggle Competition – did not do "leakage" requirement

What can you do with R?

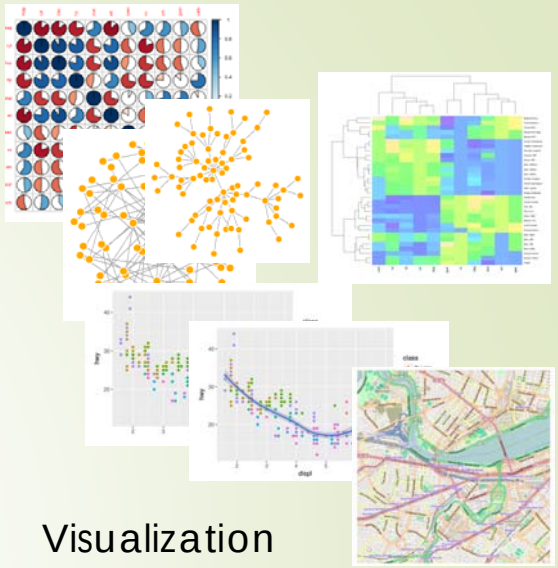


Data Manipulation, conversion, tidy

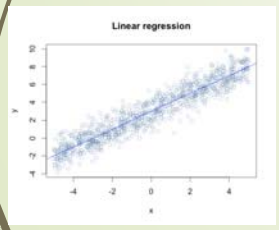


Statistics

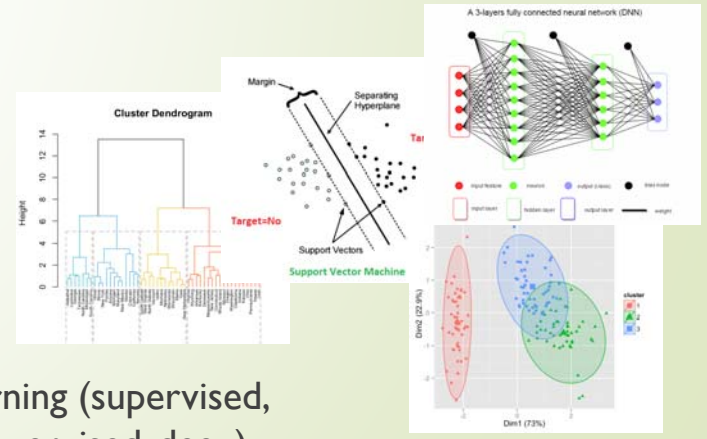
Programming Language



Visualization



Modeling: regression (linear, logistic)



Learning (supervised, unsupervised, deep)

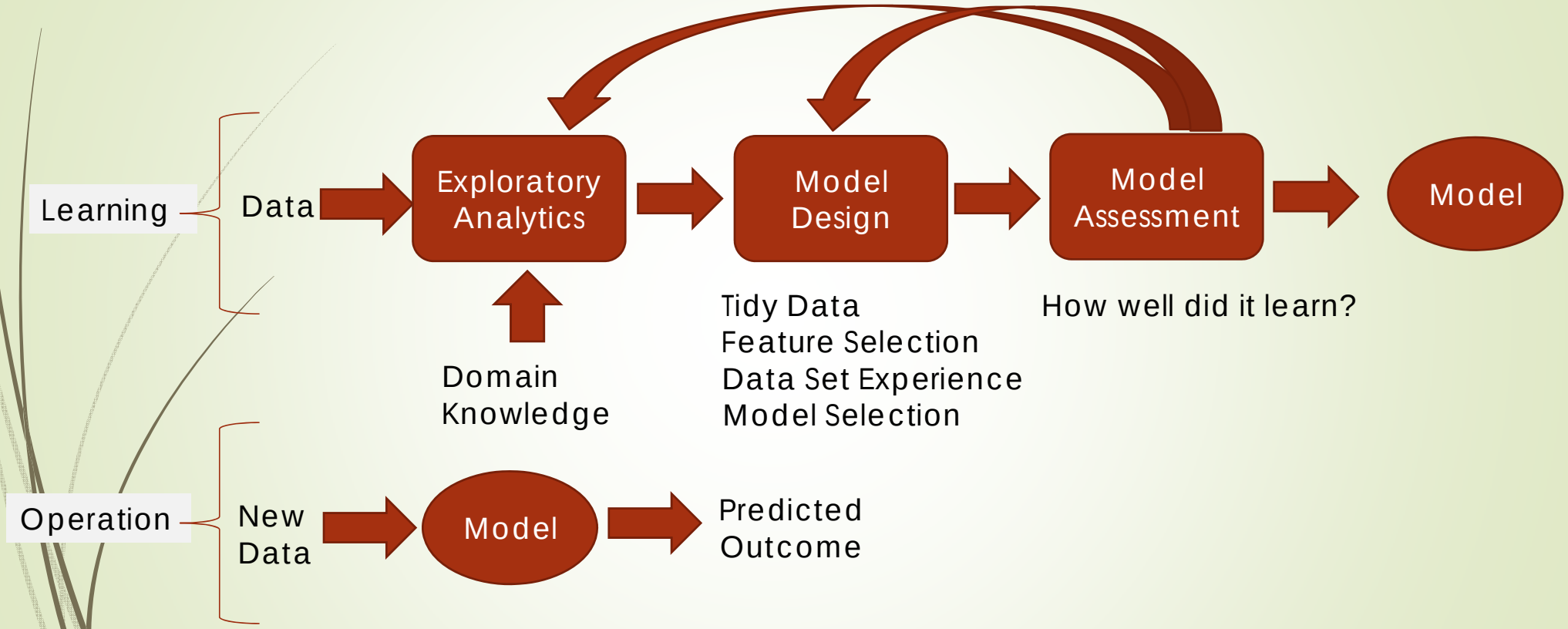
Norm Zeck

Types of analytics applications

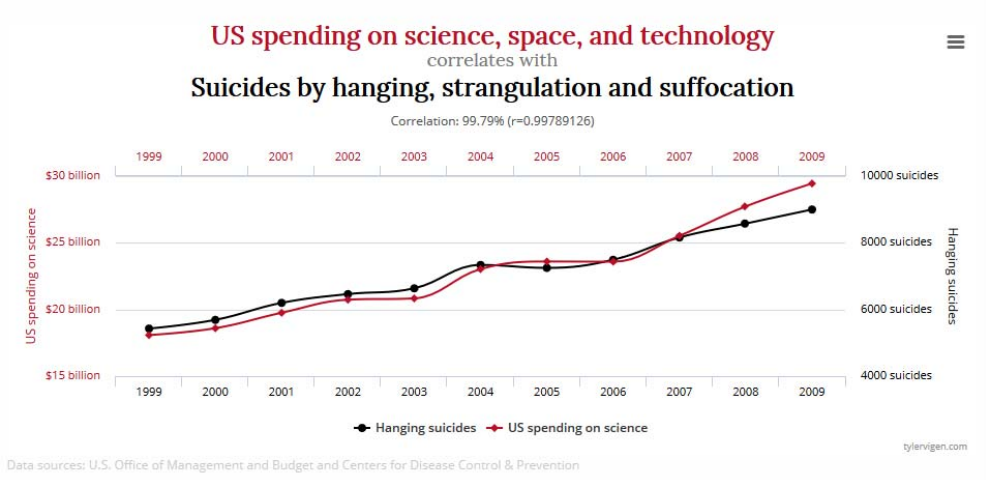
- Confusing terms: Business intelligence, machine learning, artificial intelligence, "Big data"….
- Descriptive Analytics
  - Asks: What happened? Or is happening…Reports, graphs, statistics….
  - Exploratory part of a data science project
- Predictive Analytics
  - Given new data, tells what will happen.
  - Modeling part of a data science project
- Prescriptive Analytics
  - "What should be done?" or "What can we do to make _____ happen?"
  - Beyond data science: system design including analytics – what action can you comfortably take given the confidence level of the model

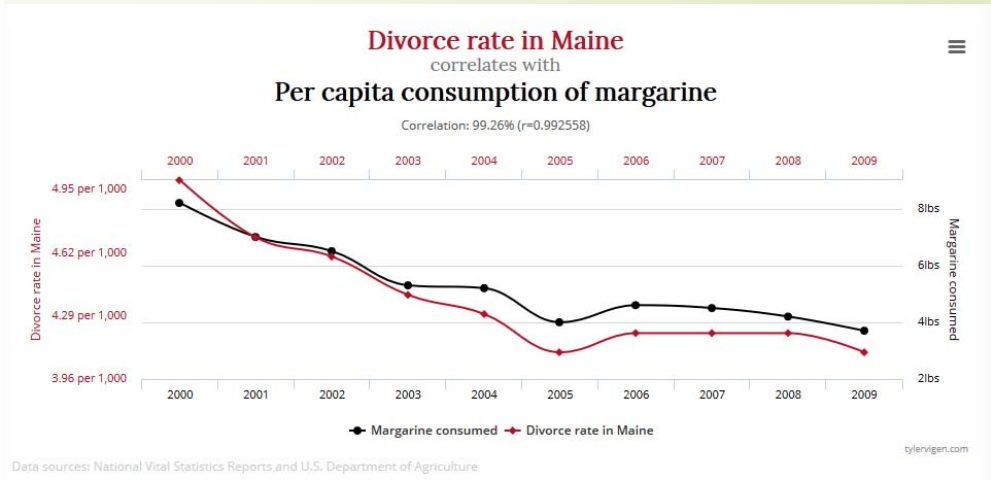
Tesla Model S Example

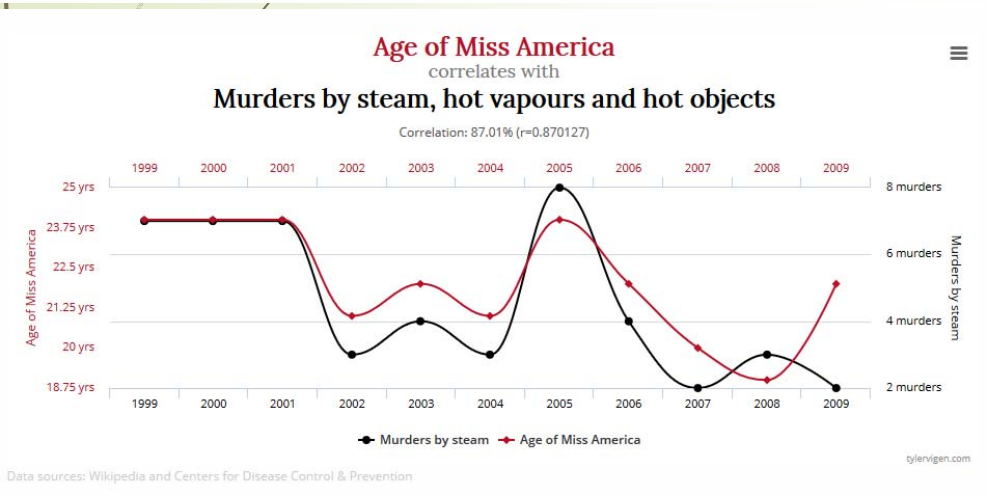
Data Science Project: Predictive Analytics

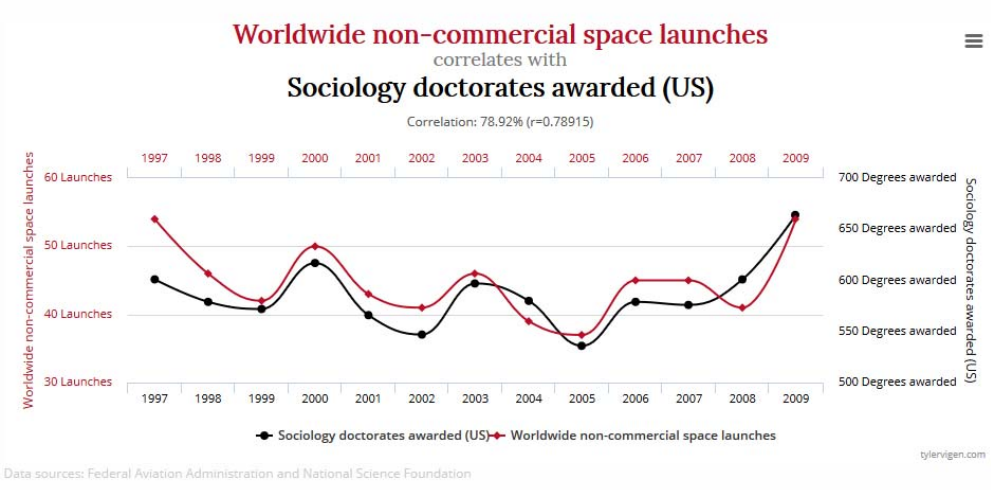


Importance of domain knowledge







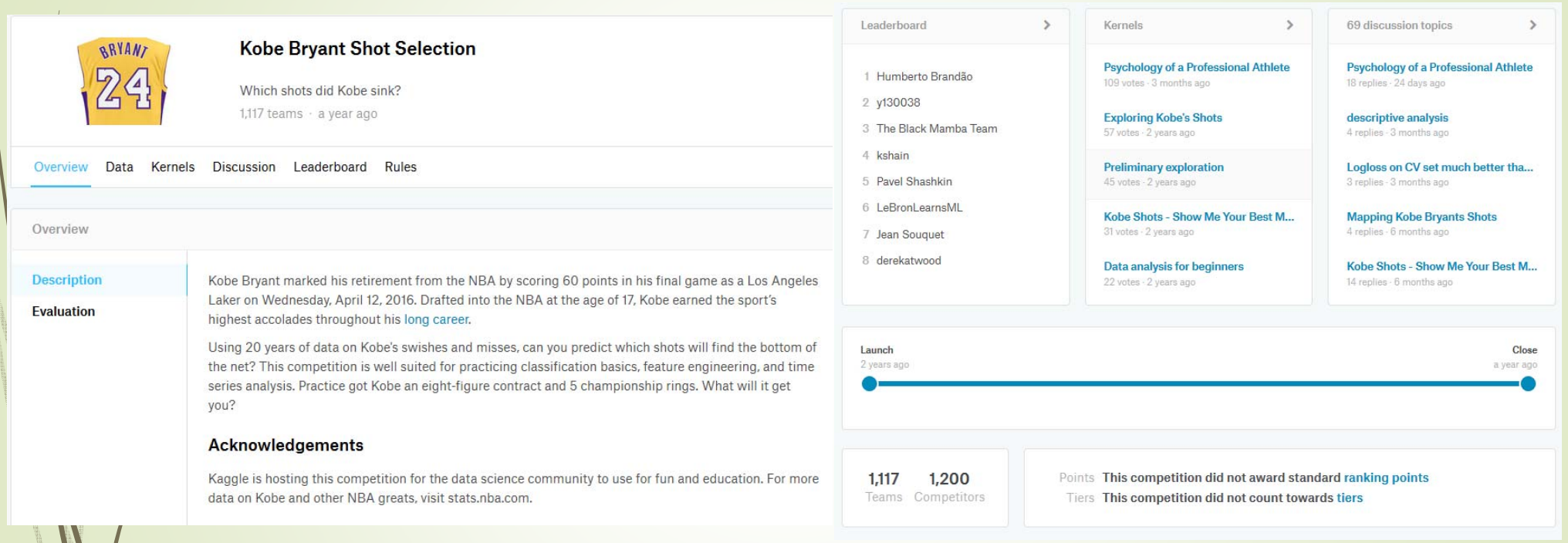


http://www.tylervigen.com/spurious-correlations

Machine Learning

- Supervised Learning
  - Given independent variables (inputs), we are **also** given the dependent variable (outcome) to use to train the model
  - Terms: "coded data set", "ground truth". We are given the answer
- Unsupervised Learning
  - Need to determine a useful outcome from the data
- Outcomes
  - Regression
    - "Continuous value". Miles per gallon for car types. House prices…. Numeric values.
  - Classification
    - Category: Is the object a person or car? Cars that are: 10 to 20 MPG, 20 to 30 MPG, 30 to 40 MPG.

Kaggle Kobe Bryant Shot Selection



CodeBook

Full Set
30,697 samples, 25 Variables
Training Set (coded)
25697 samples
Test Set (not coded)
5000 samples

Outcome →

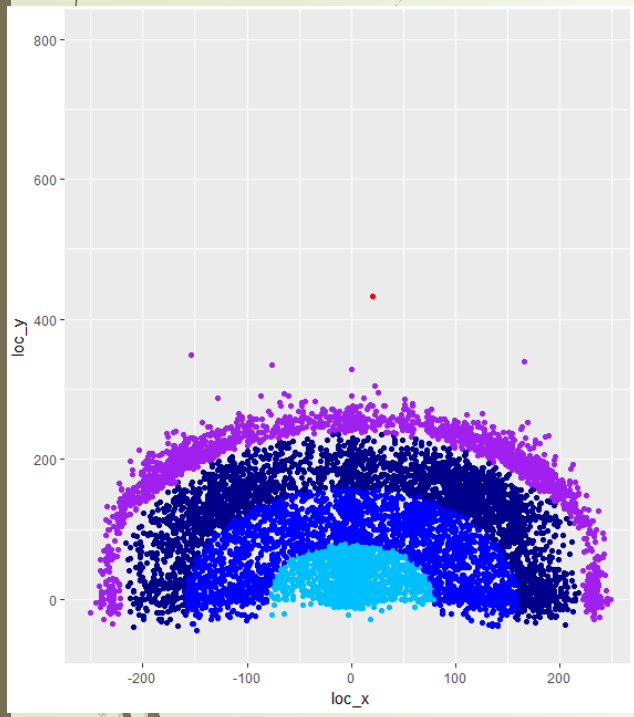
Variable	Info	Type	Grouping
season	Year span like 2000-01, 2015-16; 20 total	Categorical	Date
game_date	Date of the game	Date	Date
game_event_id	Numbered event in game	Integer	Game
game_id	Number assigned to each game	Integer	Game
playoffs	Regular or playoff game	Categorical	Game
minutes_remaining	Minutes remaining in quarter	Integer	Game Time
period	Period. Typically 1-4, but overtime 5,6,7	Categorical	Game Time
seconds_remaining	Seconds remaining in quarter	Integer	Game Time
shot_id	Sequential # for each shot	Integer	Index
lat	X location	Float	Location
loc_x	X location (0.1 ft)	Integer	Location
loc_y	Y location (0.1 ft)	Integer	Location
lon	Y location	Float	Location
shot_distance	Feet from basket, 0 is valid	Integer	Location
shot_zone_area	Left, right, center...6 levels	Categorical	Location
shot_zone_basic	7 levels: Above the Break 3; Backcourt; In The Paint (Non-RA - restricted area); Left Corner 3; Right Corner 3; Mid-Range; Restricted Area;	Categorical	Location
shot_zone_range	One of 5 zones: backcourt; 24+; 16-24 ft.; 8 to 16; less than 8;	Categorical	Location
shot_made_flag	Made/miss, this is what to predict	Categorical	Outcome
action_type	Detail shot type. 57 Levels: Reverse Layup Shot; Running Jump Shot; Jump Shot; Slam Dunk Shot...	Categorical	Shot type
combined_shot_type	More general shot type, 6 levele: Bank Shot; Dunk; Hook Shot; Jump Shot; Layup; Tip Shot	Categorical	Shot type
shot_type	2 or 3 point	Categorical	Shot type
team_id	Lakers	Integer	Team
team_name	Lakers	Categorical	Team
matchup	Opponent and home vs away	Categorical	Team
opponent	Opponent team	Categorical	Team

Spatial view of the data

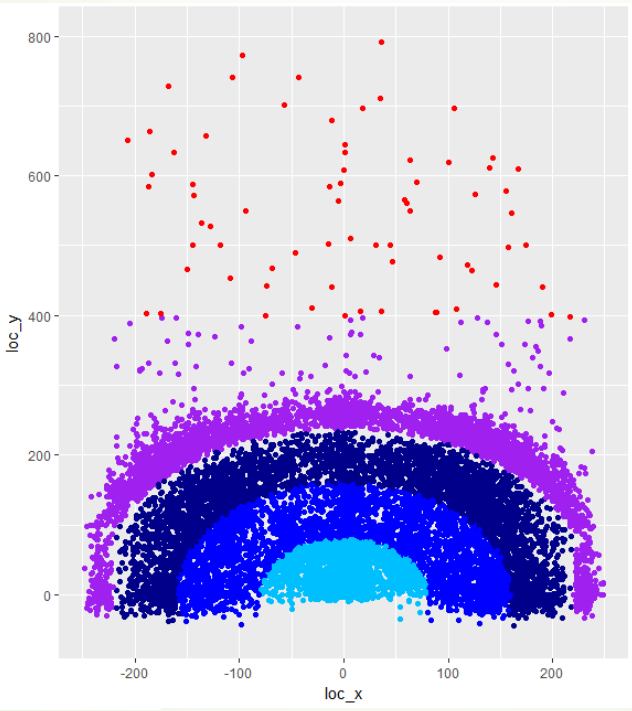
Basket ball court

Shot by distance zone

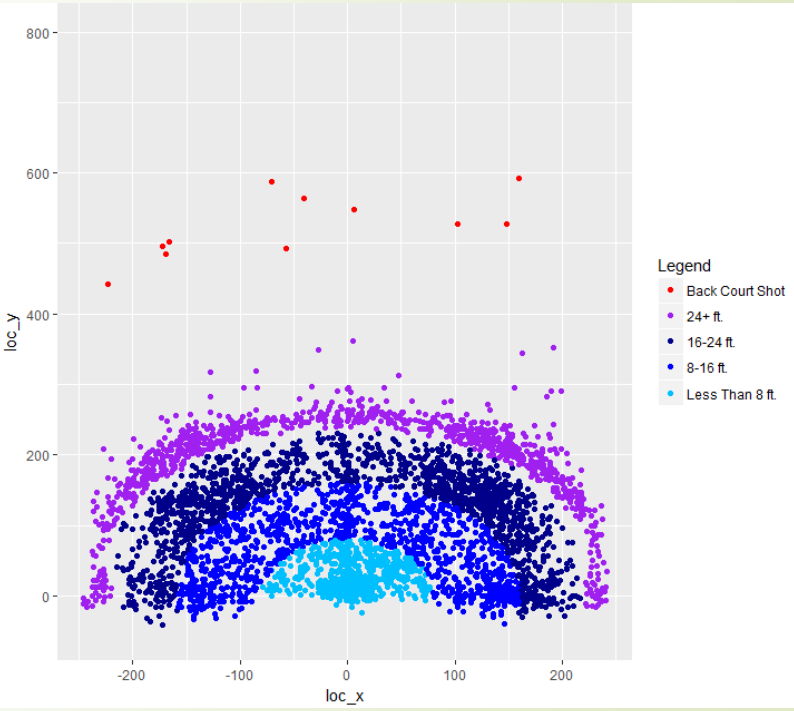
Shots Made



Shots Not Made



Test Set



Scale: 1 ~ 0.1 ft

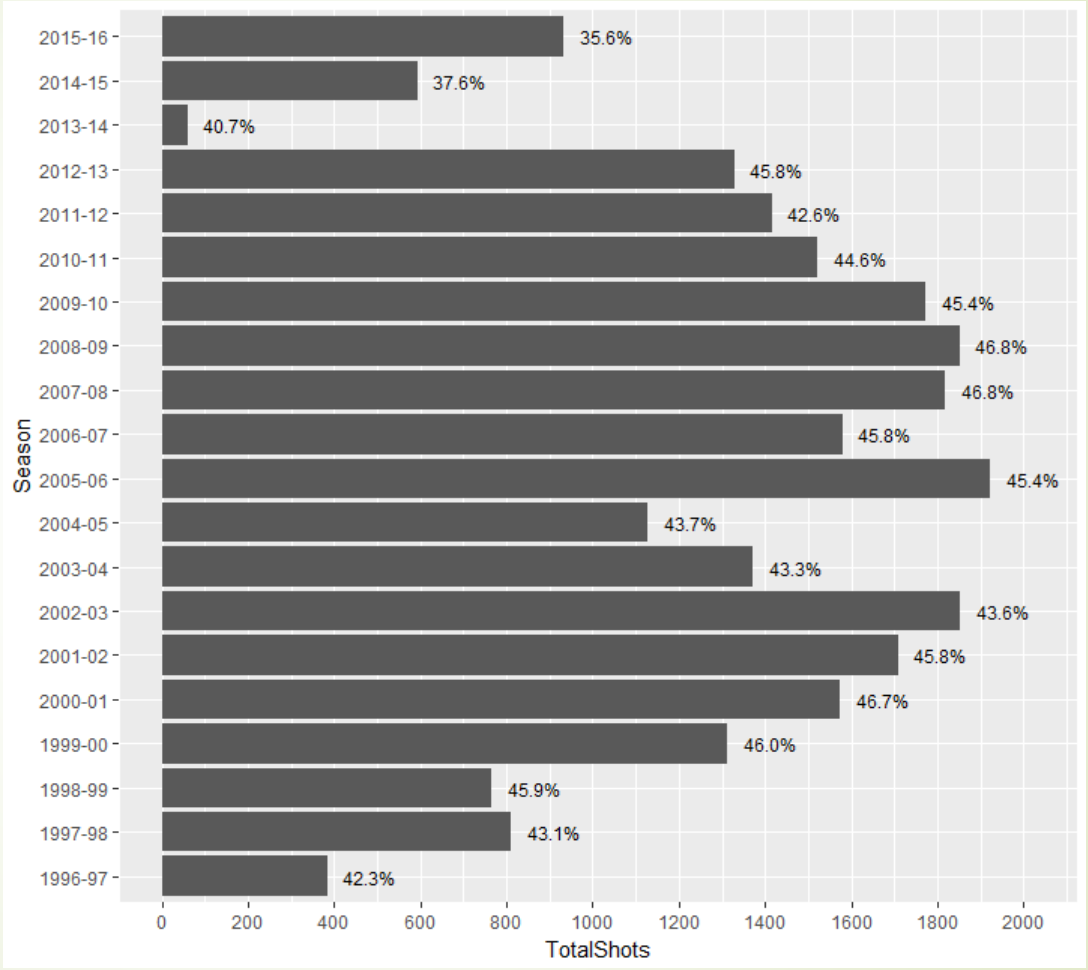
Exploratory visualization of the data

Focus on percent made, number of shots by categorical variable

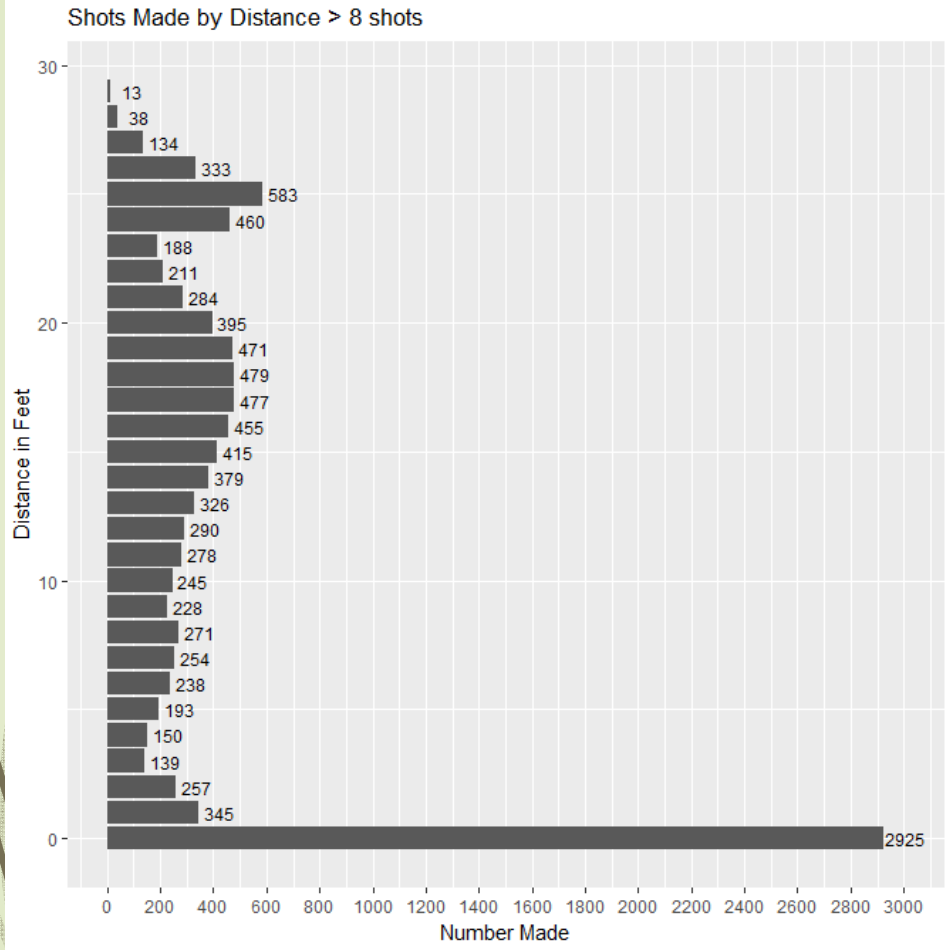
Shots by Season, Percent made

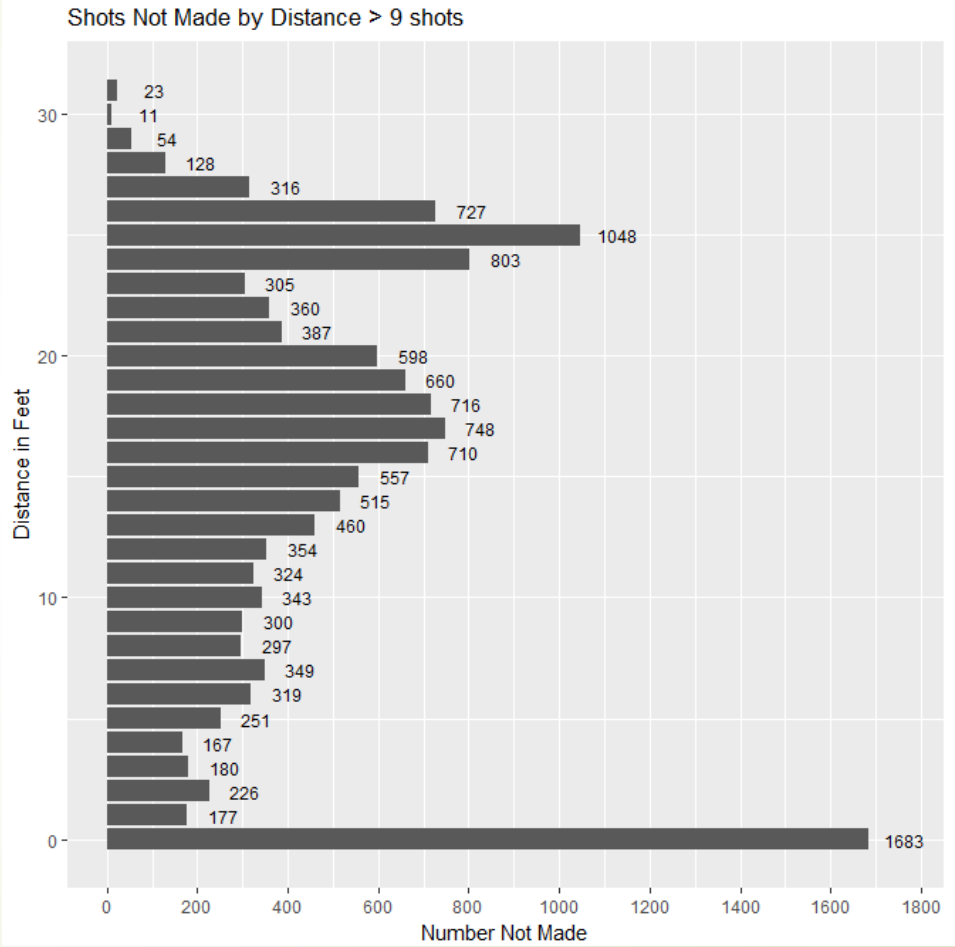
2013 Injury

Percent made ranges from 42.3% to 46.8% (4.5% delta) before injury

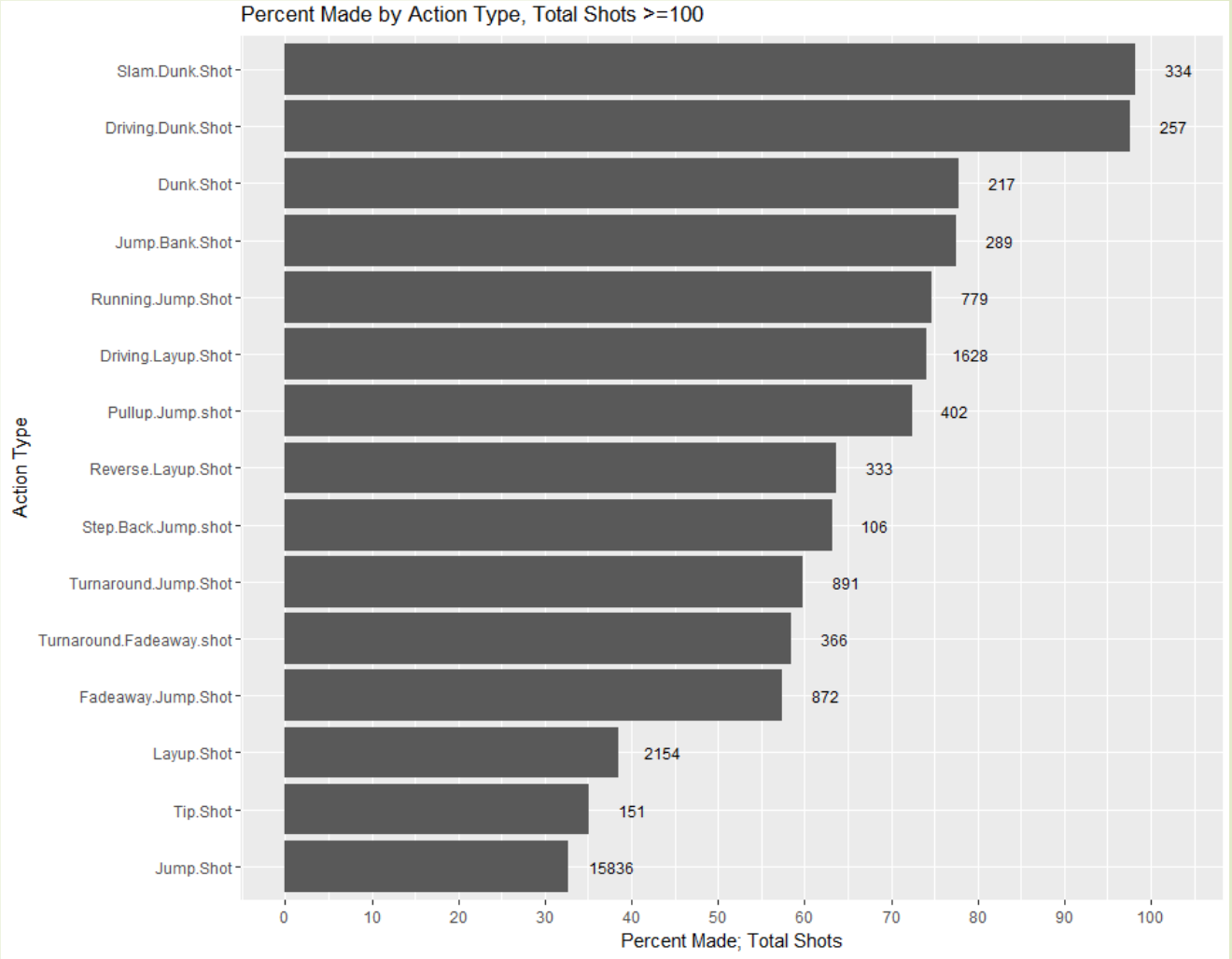


Shots by Distance

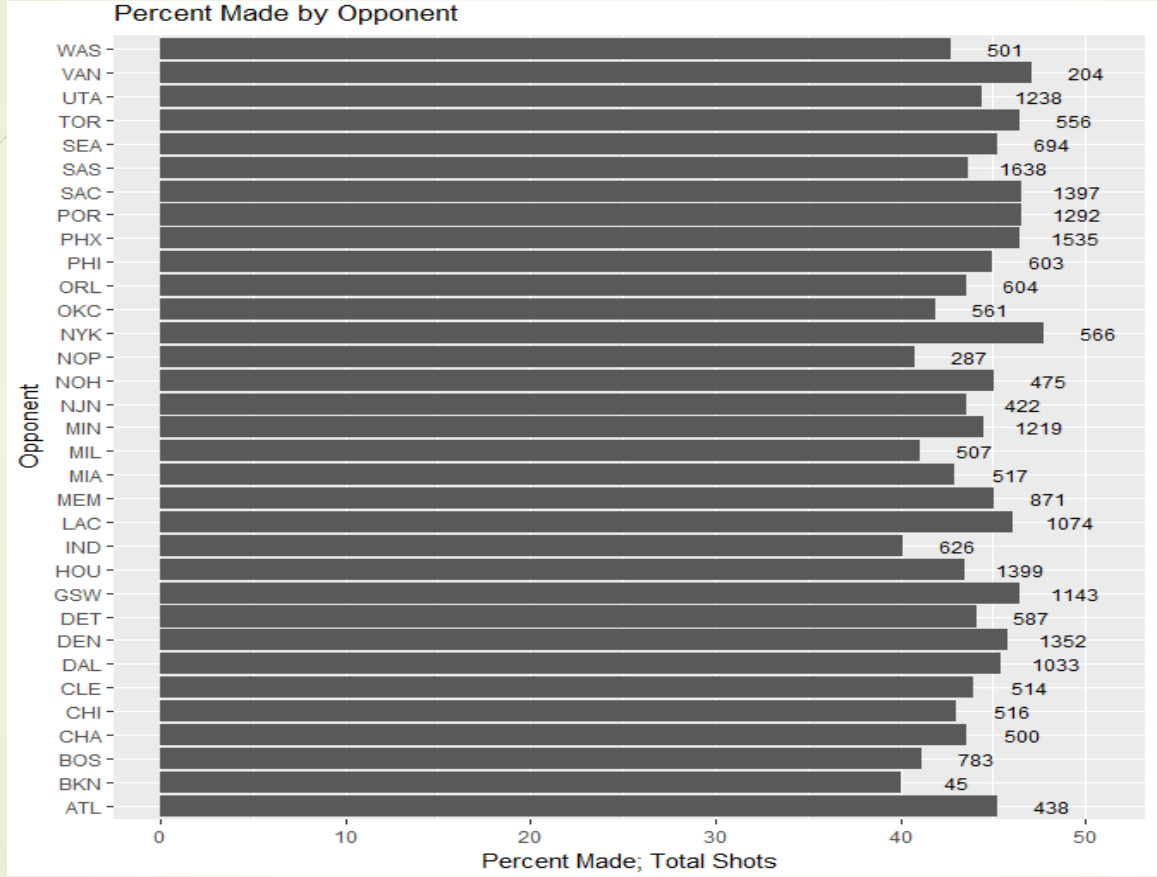




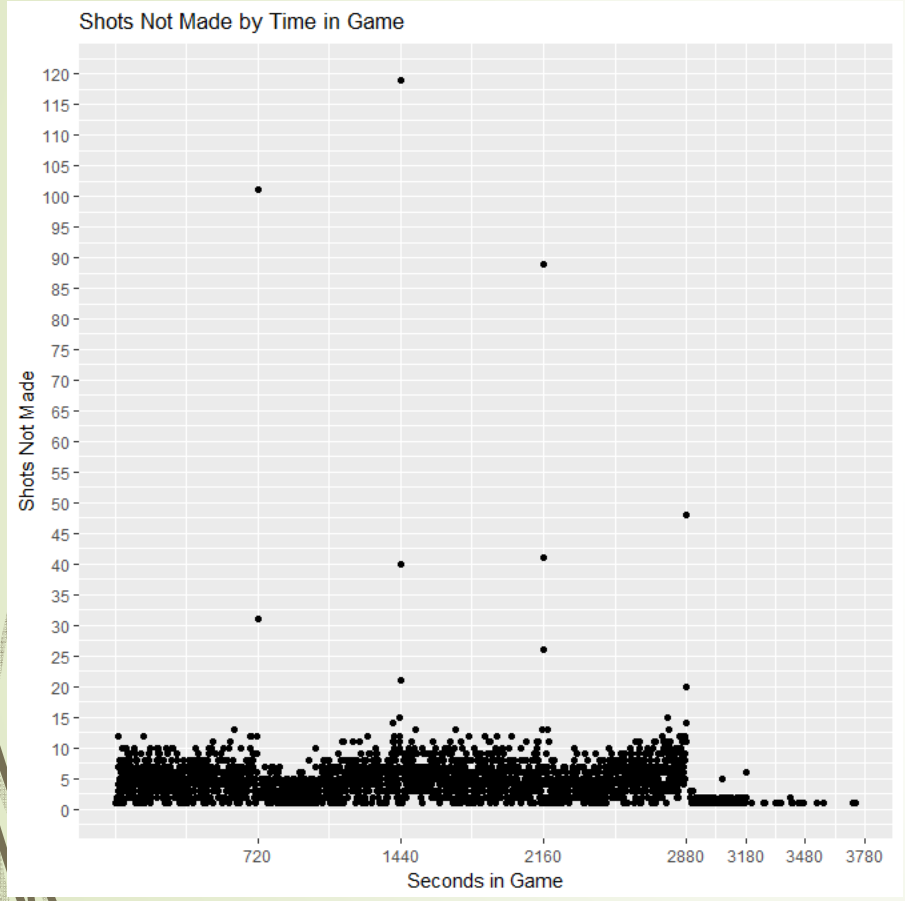
Action Type > 100 shots

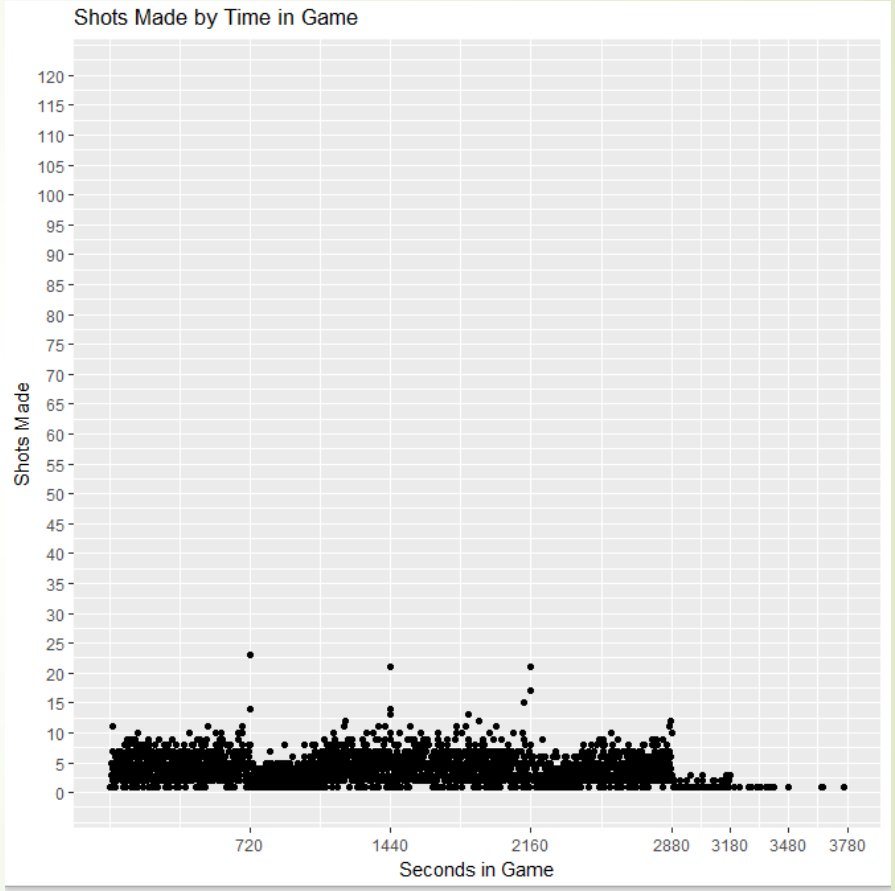


Shots by Opponent, Percent Made



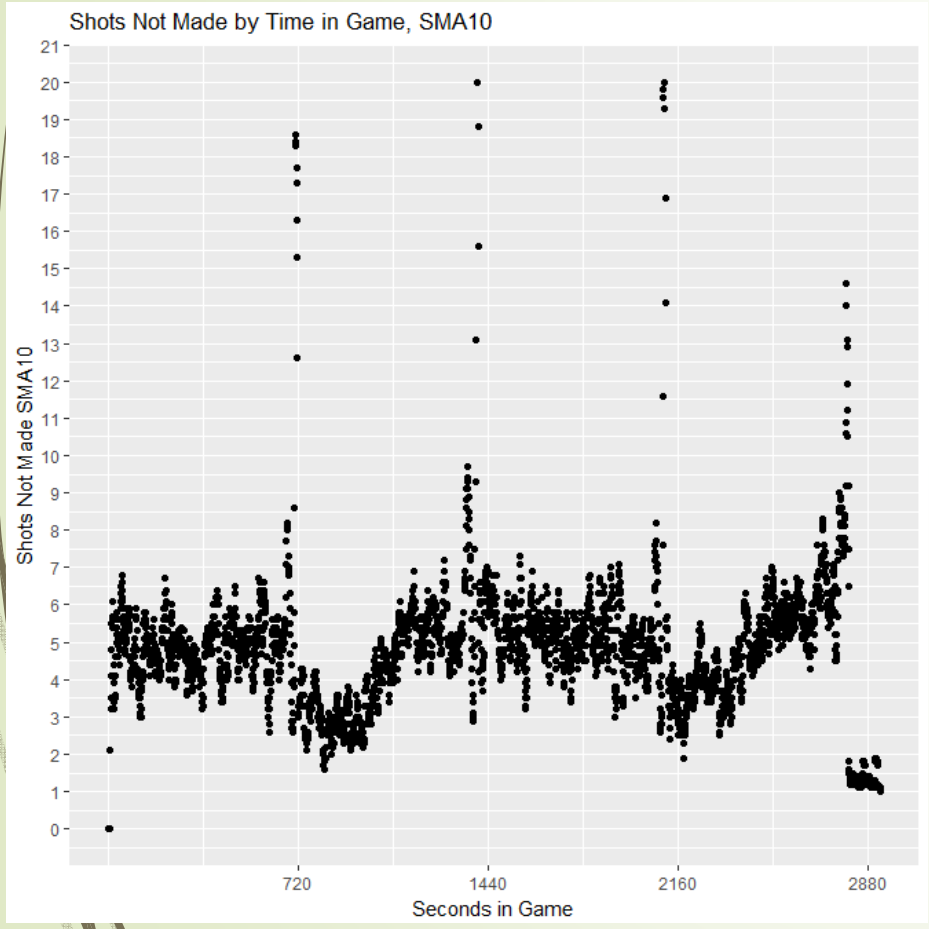
Shots by time in game

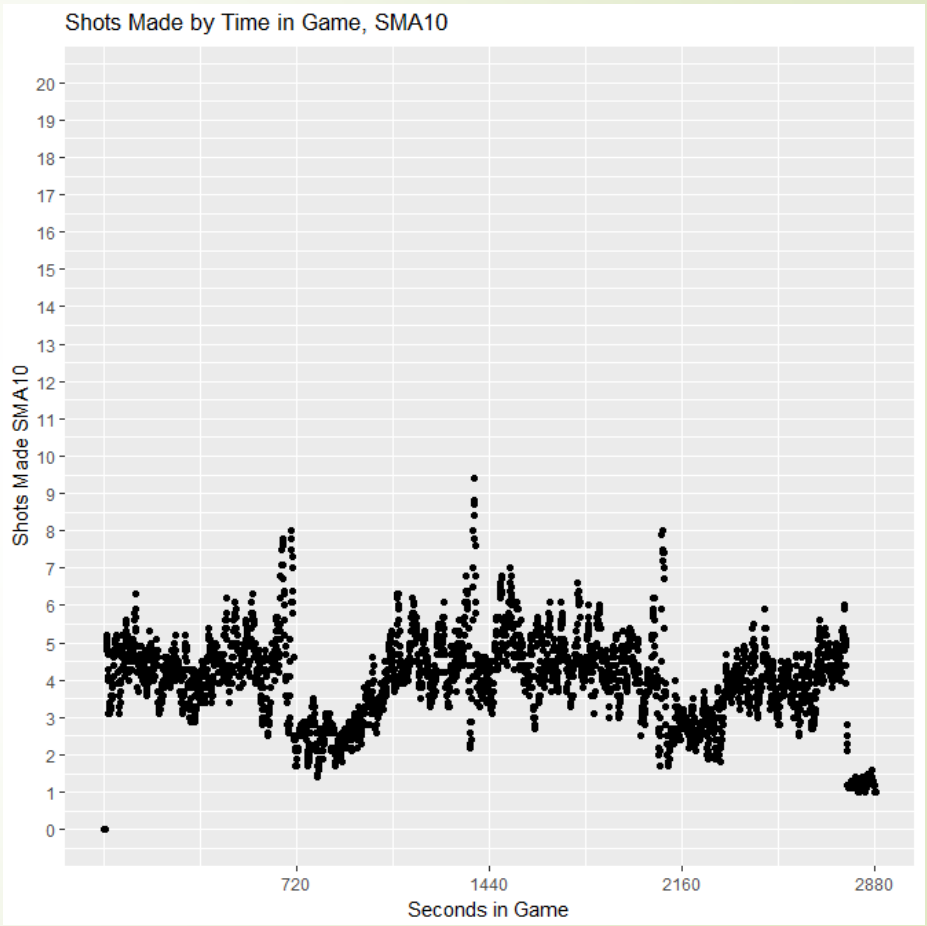




Shots by time in game

Simple moving average, 10 seconds





Which Variables to Use for Prediction?

Build on learnings from visualization & analysis

Chosen Variable Set

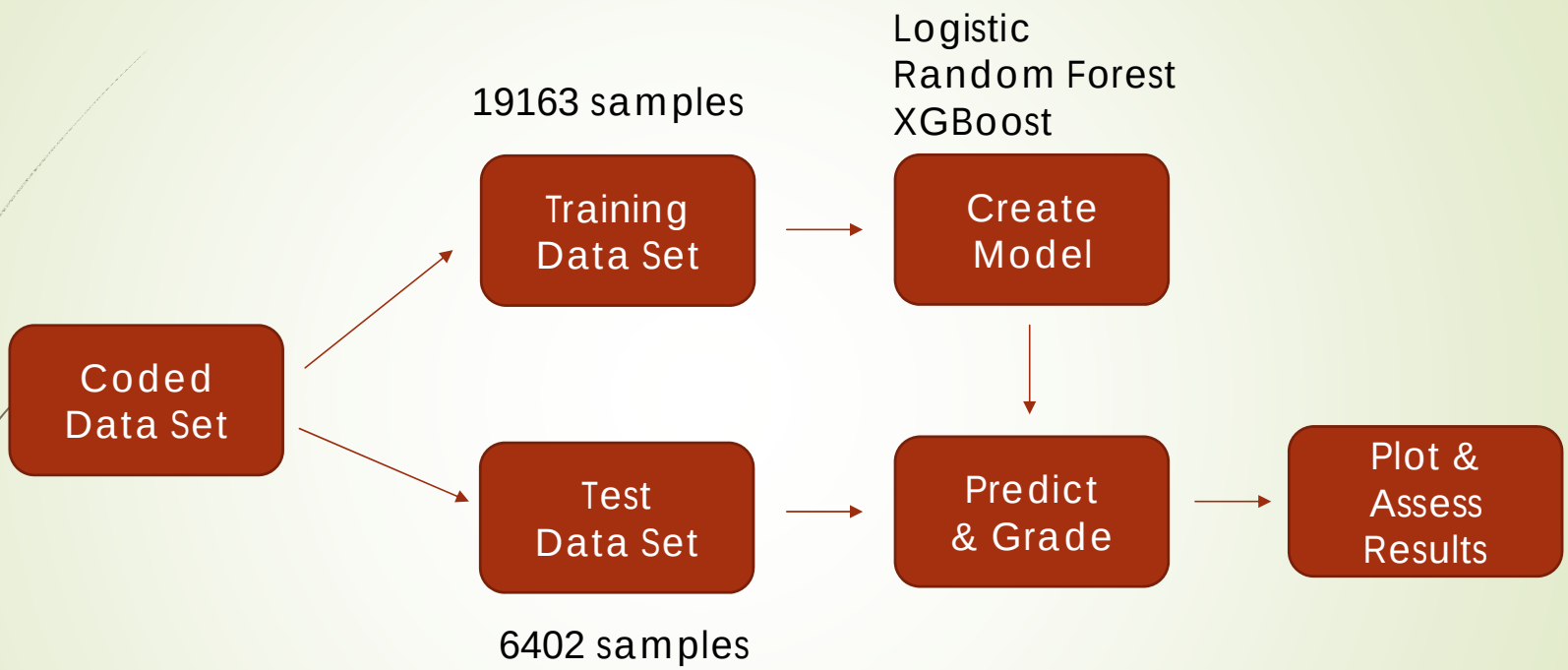
Variable	Info	Type	Grouping	Prediction
season	Year span like 2000-01, 2015-16; 20 total	Categorical	Date	Y
game_date	*Date of the game*	*Date*	*Date*	*N*
game_event_id	*Numbered event in game*	*Integer*	*Game*	*N*
game_id	Number assigned to each game	Integer	Game	Y
playoffs	*Regular or playoff game*	*Categorical*	*Game*	*N*
minutes_remaining	*Minutes remaining in quarter*	*Integer*	*Game Time*	*N*
period	*Period. Typically 1-4, but overtime 5,6,7*	*Categorical*	*Game Time*	*N*
seconds_remaining	*Seconds remaining in quarter*	*Integer*	*Game Time*	*N*
shot_id	*Sequential # for each shot*	*Integer*	*Index*	*N*
lat	*X location*	*Float*	*Location*	*N*
loc_x	X location (0.1 ft)	Integer	Location	Y
loc_y	Y location (0.1 ft)	Integer	Location	Y
lon	*Y location*	*Float*	*Location*	*N*
shot_distance	Feet from basket, 0 is valid	Integer	Location	Y
shot_zone_area	Left, right, center...6 levels	Categorical	Location	Y
shot_zone_basic	7 levels: Above the Break 3; Backcourt; In The Paint (Non-RA - restricted area); Left Corner 3; Right Corner 3; Mid-Range; Restricted Area;	Categorical	Location	Y
shot_zone_range	One of 5 zones: backcourt; 24+; 16-24 ft.; 8 to 16; less than 8;	Categorical	Location	Y
shot_made_flag	Made/miss, this is what to predict	Categorical	Outcome	Y
action_type	Detail shot type. 57 Levels: Reverse Layup Shot; Running Jump Shot; Jump Shot; Slam Dunk Shot...	Categorical	Shot type	Y
combined_shot_type	*More general shot type, 6 levele: Bank Shot; Dunk; Hook Shot; Jump Shot; Layup; Tip Shot*	*Categorical*	*Shot type*	*N*
shot_type	2 or 3 point	Categorical	Shot type	Y
team_id	*Lakers*	*Integer*	*Team*	*N*
team_name	*Lakers*	*Categorical*	*Team*	*N*
matchup	*Opponent and home vs away*	*Categorical*	*Team*	*N*
opponent	*Opponent team*	*Categorical*	*Team*	*N*
game_time	Seconds in the game	Float	Game Time	Y
game_pct	Percent made for each game	Float	Game	Y
shots_made_by_second	Number of shots made by second in the game	Float	Game	Y
shots_not_made_by_second	Number of shots not made by second in the game	Float	Game	Y

Added Variables

Predictions

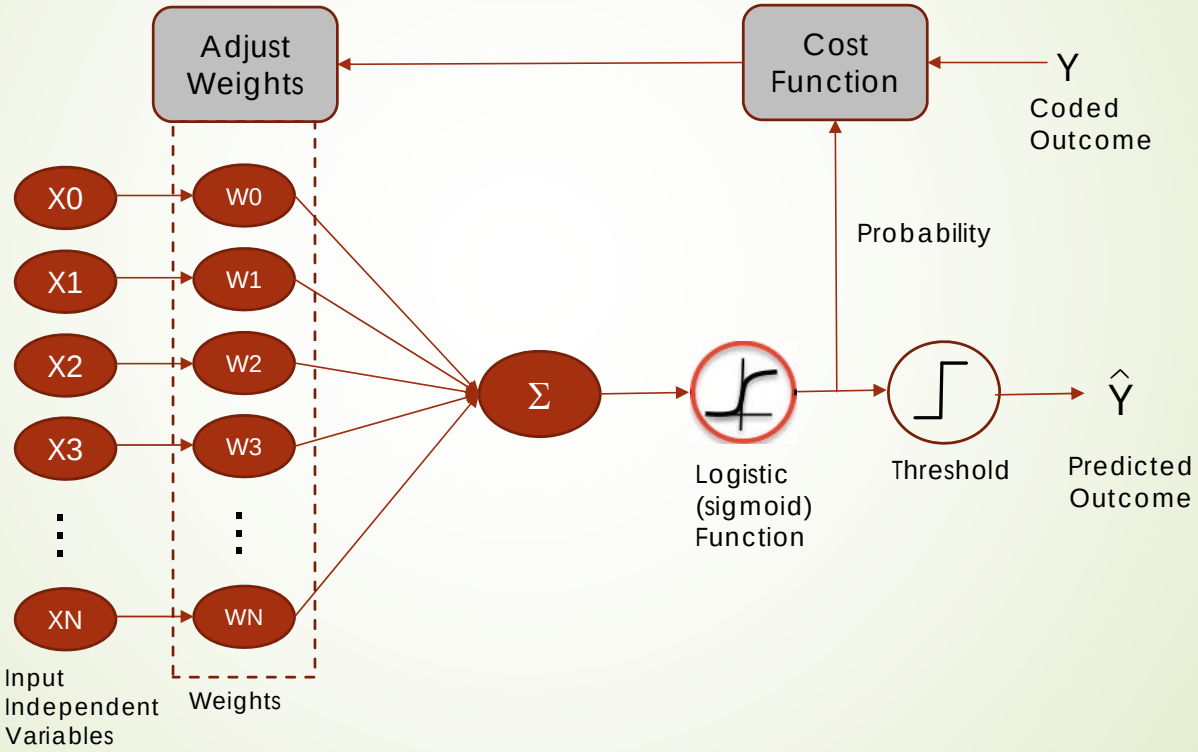
Logistic, Random Forest (Caret), XGBoost, Also tested randomForest

Modeling Process



Logistic Regression

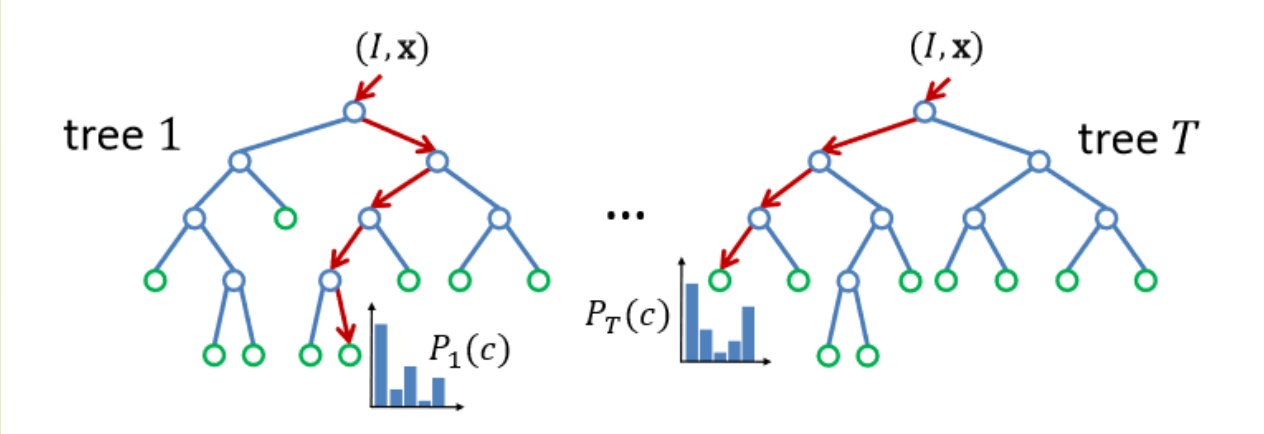
Binary classification from a continuous probability

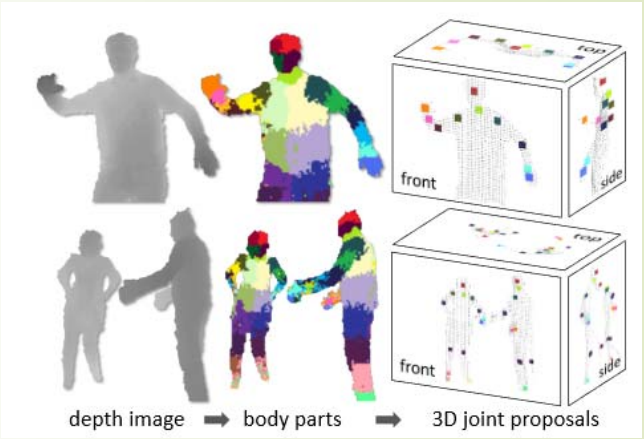


Decision trees

Example: Microsoft Kinect

Random Forest
Boosted Trees (XGBoost)





Real-Time Human Pose Recognition in Parts from Single Depth Images

https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/BodyPartRecognition.pdf

Model Setup

Data Partitioning

Caret (Classification and Regression Training) random forest example

```
set.seed(73)  #sheldon :)
# Generate a vector of random sample indexes
in_train <- createDataPartition(y=kobe_filter$shot_made_flag,
                                p=0.75,  list = FALSE)

kobe_train <- kobe_filter[in_train,]  #training set
kobe_test <- kobe_filter[-in_train,]  #test set
```

Model Optimization

Parameters and variables

Caret (Classification and Regression Training) random forest example

```
ctrl <- trainControl(method = "repeatedcv", number = 10, repeats = 3, search = "random")

model_random_forest <- train(shot_made_flag ~ season + shot_distance +
                             shot_type + shot_zone_basic + shot_zone_area +
                             loc_x + loc_y + game_time + action_type +
                             shots_notmade_by_second + shots_made_by_second + game_pct,
                       metric = "Accuracy",
                       trControl = ctrl, tuneLength = 15,
                       data=kobe_train, method = "rf",
                       ntree = 50, importance=TRUE, na.action=na.omit )
```

Model Usage & Validation

Caret (Classification and Regression Training) random forest example

```
# predict the test set
# predict generates a probability for made/not made
# we then threshold by 0.5 to generate made/not made
preds <- predict(model_random_forest, newdata=kobe_test, type="prob")
preds_df <- preds[[1]]
names(preds_df) <- c("miss", "made")
preds_threshold <- ifelse(preds_df$made > 0.5,1,0)
```

```
# Making the Confusion Matrix
 cm <- table(k_test_shot, preds_threshold)  # Confusion matrix
 accuracy <- (cm[1,1] + cm[2,2]) / (cm[1,1] + cm[2,2] + cm[1,2] + cm[2,1])
```

How to assess prediction performance?

Confusion Matrix & Accuracy

		y Predicted	
		0	1
y Actual	0	True Negative	False Positive
	1	False Negative	True Positive

$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

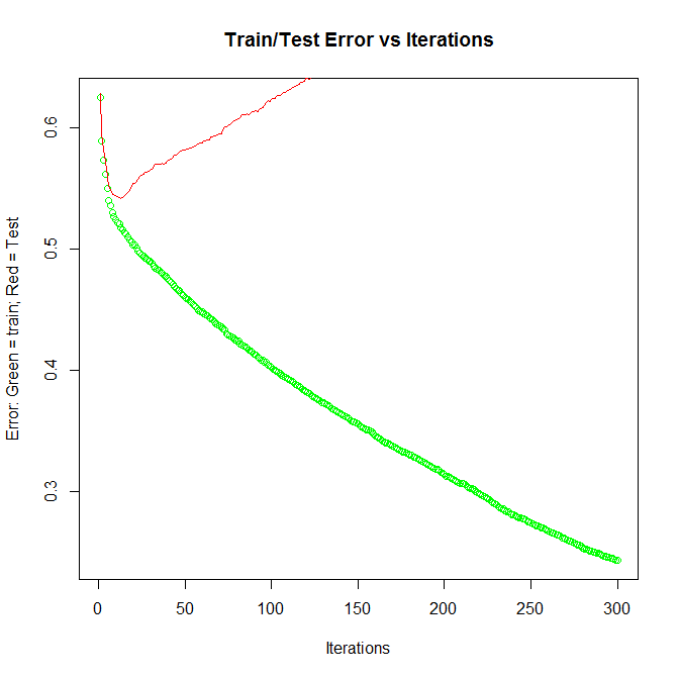
Confusion Matrix & Accuracy

	Logistic		xgboost		random forest (caret)		randomForest	
	0	1	0	1	0	1	0	1
0	2875	666	2896	645	2921	620	3151	389
1	1171	1690	1081	1780	1206	1655	2038	816
Accuracy	71.3%		73.0%		71.5%		62.0%	
Time (sec)	2.39		1.14		9813		7.14	

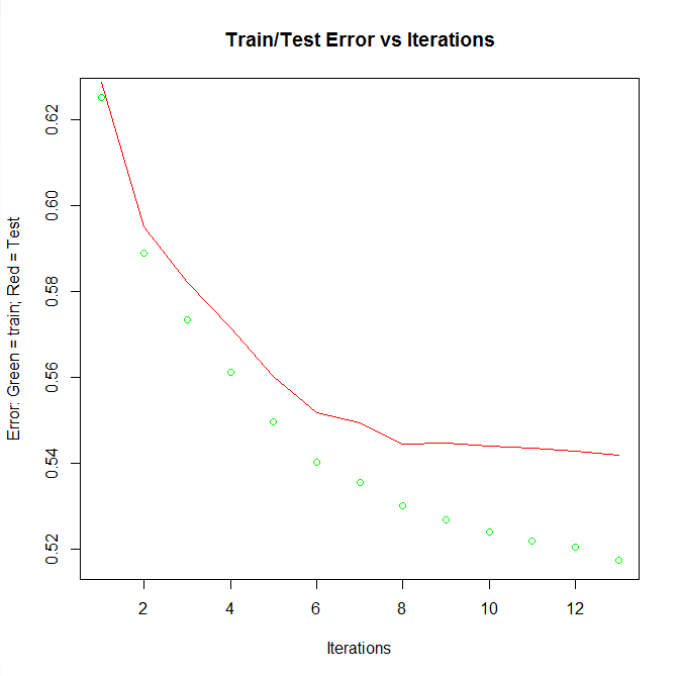
		y Predicted	
		0	1
y Actual	0	True Negative	False Positive
	1	False Negative	True Positive

9813 seconds = 2.7 hrs

XGBoost Parameter Tuning



Eta=0.35, ltr=300
Accuracy=66.2%



Eta=0.35, ltr=13
Accuracy=72.5%

```
xg_error[xg_error$test_logloss == min(xg_error$test_logloss),]

      iter  train_logloss  test_logloss
13    13      0.517244     0.541837
```

Variable Importance

Feature Engineering

Logistic Regression

	Estimate	Std. Error	z value	Pr(>|z|)	
(Intercept)	-1.08E+01	1.44E+02	-0.075	0.94028	
action_typeDriving.Jump.shot	-2.95E+00	7.35E-01	-4.02	5.83E-05	***
action_typeFadeaway.Jump.Shot	-2.30E+00	5.50E-01	-4.178	2.94E-05	***
action_typeHook.Shot	-3.20E+00	6.18E-01	-5.183	2.19E-07	***
action_typeJump.Shot	-3.46E+00	5.44E-01	-6.374	1.85E-10	***
action_typeLayup.Shot	-3.11E+00	5.24E-01	-5.927	3.09E-09	***
action_typeReverse.Layup.Shot	-2.39E+00	5.39E-01	-4.438	9.07E-06	***
action_typeStep.Back.Jump.shot	-2.06E+00	5.95E-01	-3.466	0.000528	***
action_typeTip.Shot	-3.33E+00	5.62E-01	-5.927	3.09E-09	***
action_typeTurnaround.Fadeaway.shot	-2.01E+00	5.60E-01	-3.589	0.000332	***
action_typeTurnaround.Jump.Shot	-2.27E+00	5.49E-01	-4.124	3.72E-05	***
shots_notmade_by_second	-7.34E-02	3.13E-03	-23.422	< 2e-16	***
shots_made_by_second	2.31E-01	7.76E-03	29.753	< 2e-16	***
game_pct	4.25E-02	1.54E-03	27.687	< 2e-16	***
shot_distance	2.36E-02	8.89E-03	2.659	0.007845	**
action_typeAlley.Oop.Layup.shot	-1.78E+00	6.12E-01	-2.903	0.003692	**
action_typeDriving.Hook.Shot	-2.12E+00	8.18E-01	-2.59	0.009611	**
action_typeDriving.Layup.Shot	-1.61E+00	5.26E-01	-3.056	0.00224	**
action_typeDunk.Shot	-1.48E+00	5.56E-01	-2.65	0.008043	**
action_typeFinger.Roll.Shot	-2.31E+00	7.19E-01	-3.213	0.001312	**
action_typePullup.Bank.shot	-2.28E+00	8.57E-01	-2.655	0.007924	**
action_typePullup.Jump.shot	-1.62E+00	5.62E-01	-2.877	0.004018	**
action_typeRunning.Jump.Shot	-1.43E+00	5.52E-01	-2.594	0.009478	**
action_typeRunning.Layup.Shot	-1.84E+00	6.23E-01	-2.958	0.003098	**
action_typeTurnaround.Hook.Shot	-2.51E+00	9.64E-01	-2.6	0.009321	**
shot_zone_basicLeft Corner 3	4.00E-01	1.99E-01	2.011	0.044353	*
action_typeDriving.Finger.Roll.Shot	-1.27E+00	6.35E-01	-2.001	0.045363	*
action_typeDriving.Reverse.Layup.Shot	-1.42E+00	6.00E-01	-2.367	0.017939	*
action_typeFinger.Roll.Layup.Shot	-1.62E+00	8.00E-01	-2.024	0.042991	*
action_typeFloating.Jump.shot	-1.49E+00	6.07E-01	-2.445	0.014487	*
action_typeJump.Bank.Shot	-1.30E+00	5.71E-01	-2.279	0.022687	*
action_typePutback.Dunk.Shot	-3.16E+00	1.53E+00	-2.066	0.038864	*
action_typeJump.Hook.Shot	-1.56E+00	8.60E-01	-1.809	0.070414	.
action_typePutback.Layup.Shot	-1.91E+00	1.03E+00	-1.862	0.062568	.
action_typeTurnaround.Bank.shot	-1.26E+00	6.69E-01	-1.88	0.060097	.

```
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

XGBoost

	Feature	Gain	Cover	Frequency
1	action_type	0.28415	0.20674	0.08803
2	game_pct	0.16535	0.17451	0.15479
3	shots_made_by_second	0.16297	0.19144	0.14528
4	shots_notmade_by_second	0.15386	0.17723	0.10840
5	shot_distance	0.05356	0.05951	0.06789
6	loc_y	0.04793	0.06090	0.10840
7	game_time	0.04733	0.04359	0.13193
8	loc_x	0.03919	0.04697	0.08984
9	season	0.02908	0.02497	0.07422
10	shot_zone_basic	0.00580	0.00570	0.01199
11	shot_zone_range	0.00536	0.00416	0.00498
12	shot_zone_area	0.00499	0.00349	0.01313
13	shot_type	0.00041	0.00080	0.00113

Random Forest (Caret)

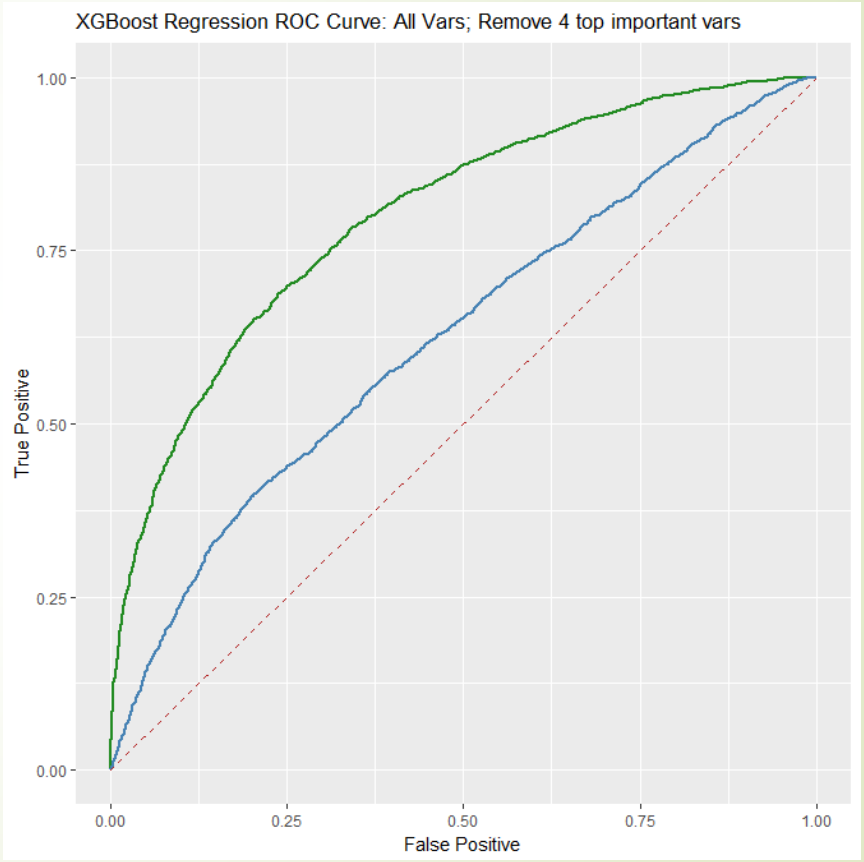
	MeanDecreaseAccuracy
shots_notmade_by_second	26.27371269
shots_made_by_second	24.3478625
action_typeJump.Shot	23.96860105
action_typeLayup.Shot	22.40470404
game_pct	22.11663757
loc_x	12.53466977
action_typeSlam.Dunk.Shot	11.86498281
loc_y	11.47755651
shot_distance	10.58029387
action_typeDriving.Dunk.Shot	10.36861343
action_typePullup.Jump.shot	7.372344572
action_typeRunning.Jump.Shot	6.258553183
action_typeTip.Shot	5.602478132

Model support and comparison

One-Hot Encoding; ROC (Receiver Operating Characteristic)

Var1	Var2	Outcome
1.20	Cat1	Out1
1.30	Cat2	Out2
5.00	Cat1	Out3
10.40	Cat4	Out4

Var1	Cat1	Cat2	Cat4	Outcome
1.20	1	0	0	Out1
1.30	0	1	0	Out2
5.00	1	0	0	Out3
10.40	0	0	1	Out4



Add one hot encoding to xgboost

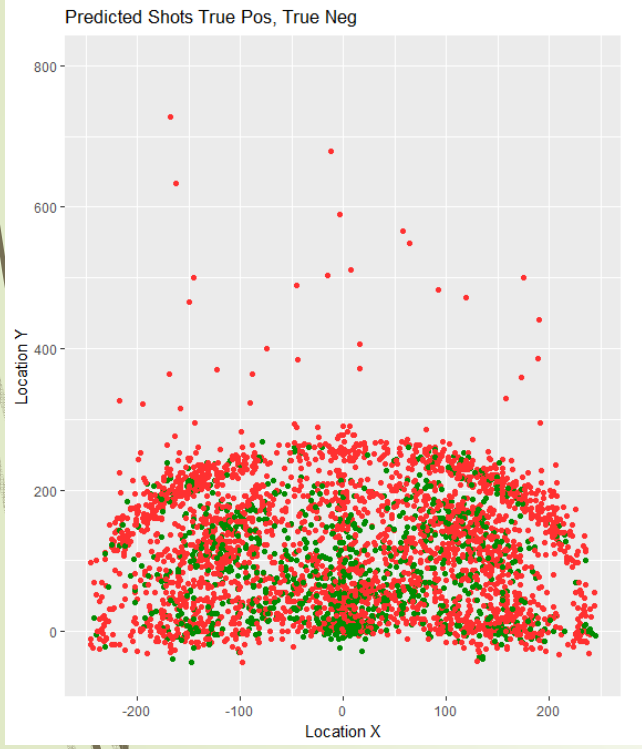
Variable Importance

Feature	Gain	Cover	Frequency
action_type.Jump.Shot	0.173663	0.090785	0.018545
shots_notmade_by_second	0.166892	0.188760	0.132192
game_pct	0.161186	0.170410	0.142891
shots_made_by_second	0.148480	0.174861	0.110556
action_type.Layup.Shot	0.062307	0.028935	0.017356
game_id	0.043759	0.042086	0.116025
game_time	0.042282	0.034741	0.112458
loc_y	0.041087	0.049272	0.082739
shot_distance	0.038701	0.041130	0.047551
loc_x	0.035813	0.039792	0.085592
season	0.020064	0.017571	0.051831
action_type.Tip.Shot	0.009189	0.012826	0.008797
action_type.Slam.Dunk.Shot	0.009096	0.027378	0.008084
action_type.Running.Jump.Shot	0.008442	0.014488	0.006419
action_type.Driving.Dunk.Shot	0.006934	0.023104	0.007133
shot_zone_area	0.006413	0.005038	0.012839
shot_zone_range	0.004214	0.003322	0.003566
shot_zone_basic	0.004096	0.002955	0.009272
action_type.Fadeaway.Jump.Shot	0.003634	0.003945	0.004517
action_type.Reverse.Layup.Shot	0.003484	0.008578	0.004755
action_type.Pullup.Jump.shot	0.002479	0.007320	0.004042
action_type.Jump.Bank.Shot	0.002110	0.004477	0.003091
action_type.Hook.Shot	0.001992	0.002994	0.002378
action_type.Turnaround.Jump.Shot	0.001235	0.001652	0.001902
action_type.Driving.Layup.Shot	0.001148	0.001114	0.001664
shot_type	0.000522	0.000290	0.001427

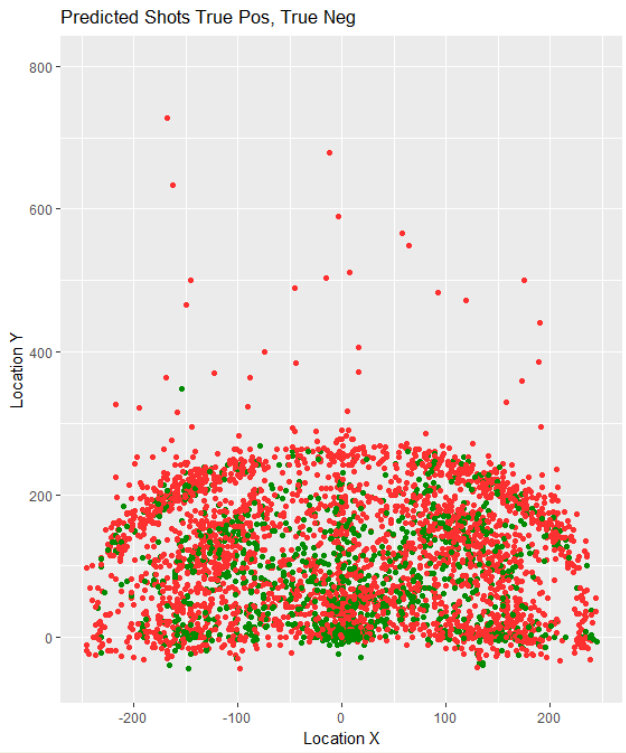
	Feature	Gain	Cover	Frequency
1	action_type	0.28415	0.20674	0.08803
2	game_pct	0.16535	0.17451	0.15479
3	shots_made_by_second	0.16297	0.19144	0.14528
4	shots_notmade_by_second	0.15386	0.17723	0.10840
5	shot_distance	0.05356	0.05951	0.06789
6	loc_y	0.04793	0.06090	0.10840
7	game_time	0.04733	0.04359	0.13193
8	loc_x	0.03919	0.04697	0.08984
9	season	0.02908	0.02497	0.07422
10	shot_zone_basic	0.00580	0.00570	0.01199
11	shot_zone_range	0.00536	0.00416	0.00498
12	shot_zone_area	0.00499	0.00349	0.01313
13	shot_type	0.00041	0.00080	0.00113

True Positive & Negative

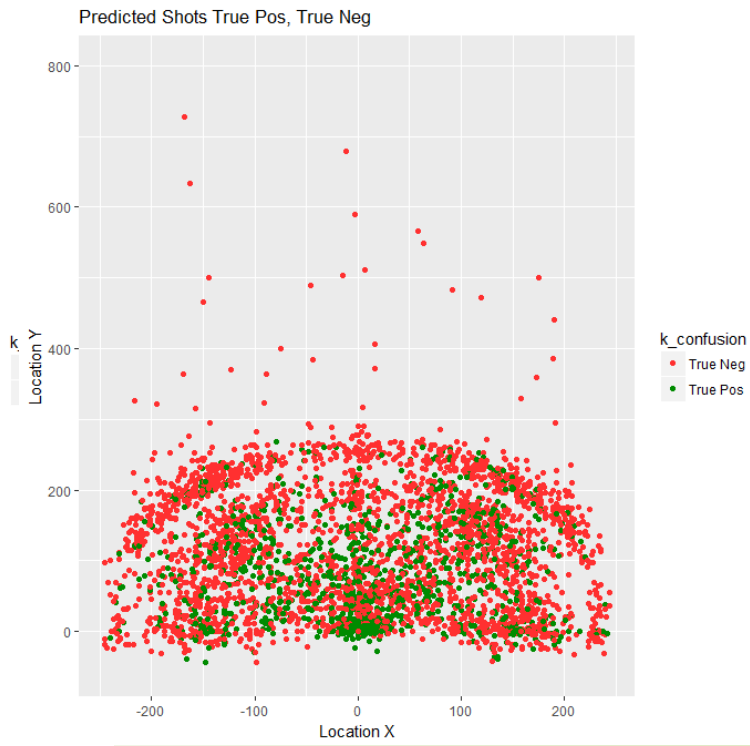
Logistic (2875, 1690)



XGBoost (2896, 1780)

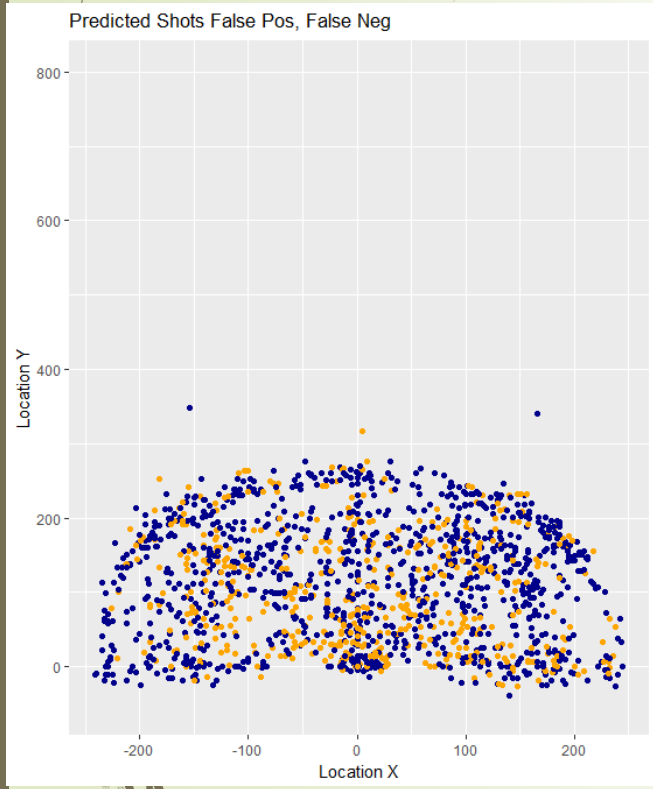


Caret (Random Forest) (2921, 1655)

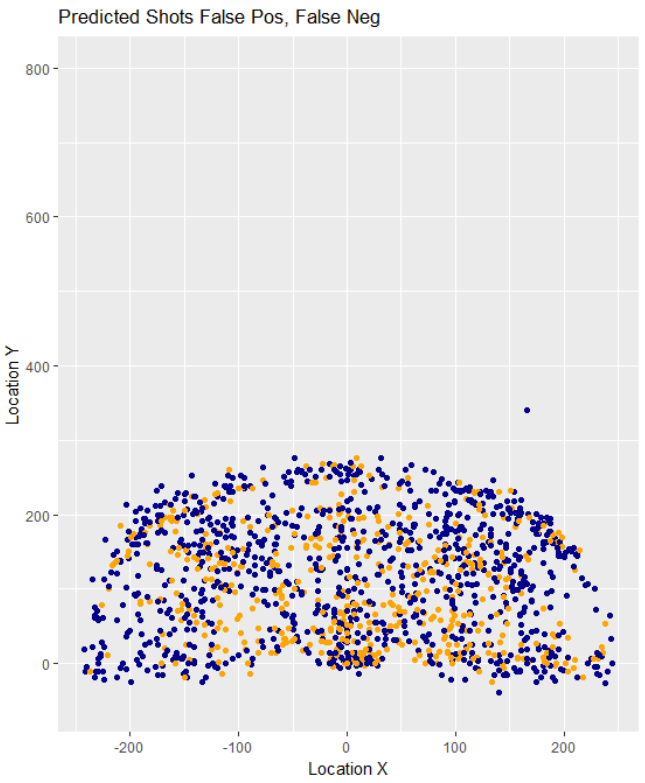


False Positive & Negative

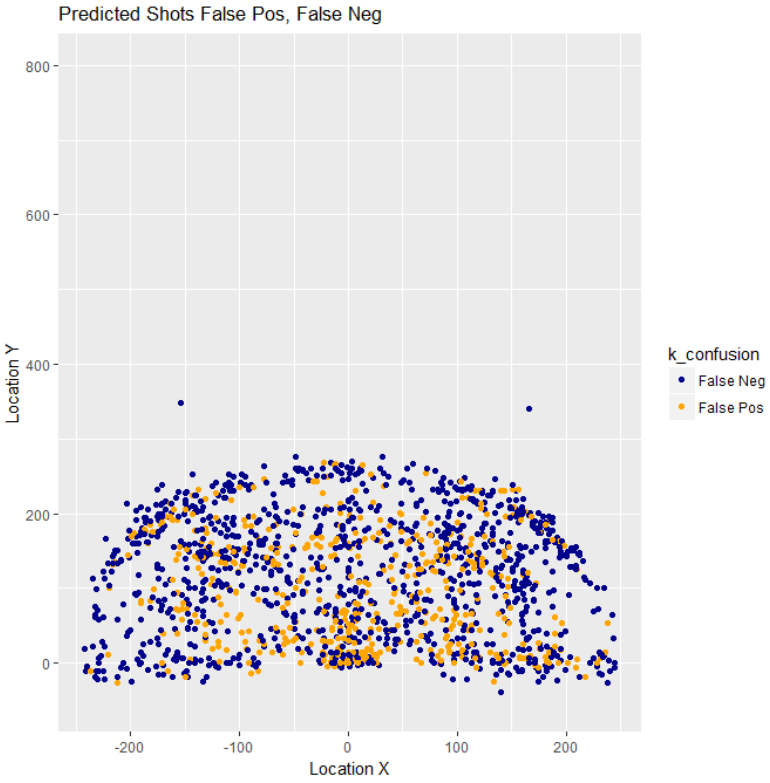
Logistic (1171, 666)



XGBoost (1081, 645)



Caret (Random Forest) (1206, 620)



Summary & Learning

- Data Science Project
  - Visualization and analysis yielded new variables that also were high importance in the models
    - Good examples for unsupervised learning
- Models
  - Given information contained in the independent variables, models topped out at ~70%
    - Models needed more information that directed successful shots (True Positive)
    - Since his percent made was from 42% to 47% there is a small real bias toward missing a shot. Models do better at predicting missed shots.
  - XGBoost. Impressive both in performance and accuracy, tuning
  - Surprised that logistic regression did as well compared to decision trees.
  - Caret random forest worked well. Used in the past.
  - Complex Algorithms: Tuning in all cases was less obvious for many parameters.

Links

- Code: https://github.com/norm42/Kobe-Bryant-Kaggle-Analysis
- xgboost
  - Paper: XGBoost: A Scalable Tree Boosting System
    - https://arxiv.org/pdf/1603.02754.pdf
  - Video
    - https://youtu.be/ufHo8vbk6g4
      - detailed but a bit slow, more on parameters
- Machine Learning: Kirill Eremenko, Udemy
  - https://www.udemy.com/machinelearning/learn/v4/overview
  - Python and R examples for each topic
- R: Kirill Eremenko, Udemy
  - R Programming A-Z™: R For Data Science
  - R Programming: Advanced Analytics In R For Data Science

Backup

Code and Data File Index

kobe_xgboost.R	XGBoost Model
kobeinit.R	Initialization and data maniplulation
kobe_explore.R	Visualization of data set
kobe_logistic.R	Logistic Model
kobe_caretrf.R	Caret Random Forest Model
kobe_func.R	Utility functions
kobeinfo-v2.xlsx	Excel file with info on the data set and model results
bbcourt.jpg	Picture of basketball court with dimensions
KobeBryant.txt	Some info on the data set from kaggle
data.csv	Data set - you need to get this from kaggle. Sign up is free, search for "Kobe" on their site will get you to the page.

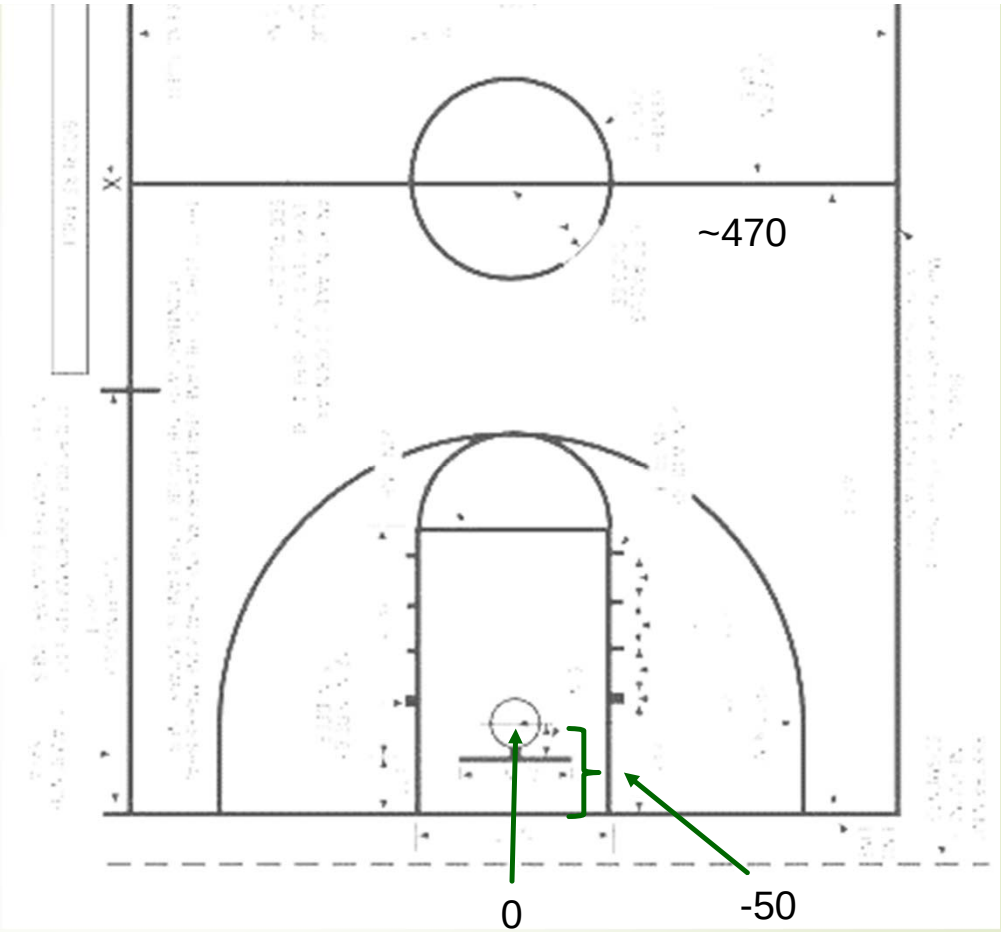
- You will have to change the "set working directory" in kobeinit.R to the location of your files.
- There is a function call commented out in kobeinit.R, check_pkgs(), that will check and install packages. It does ask first. You can source kobe_func.R first, then run check_pkgs() before getting started.
- Also, I have only run this on the windows version of R. Other than the directory name in setwd, should work under Linux.

Operation:

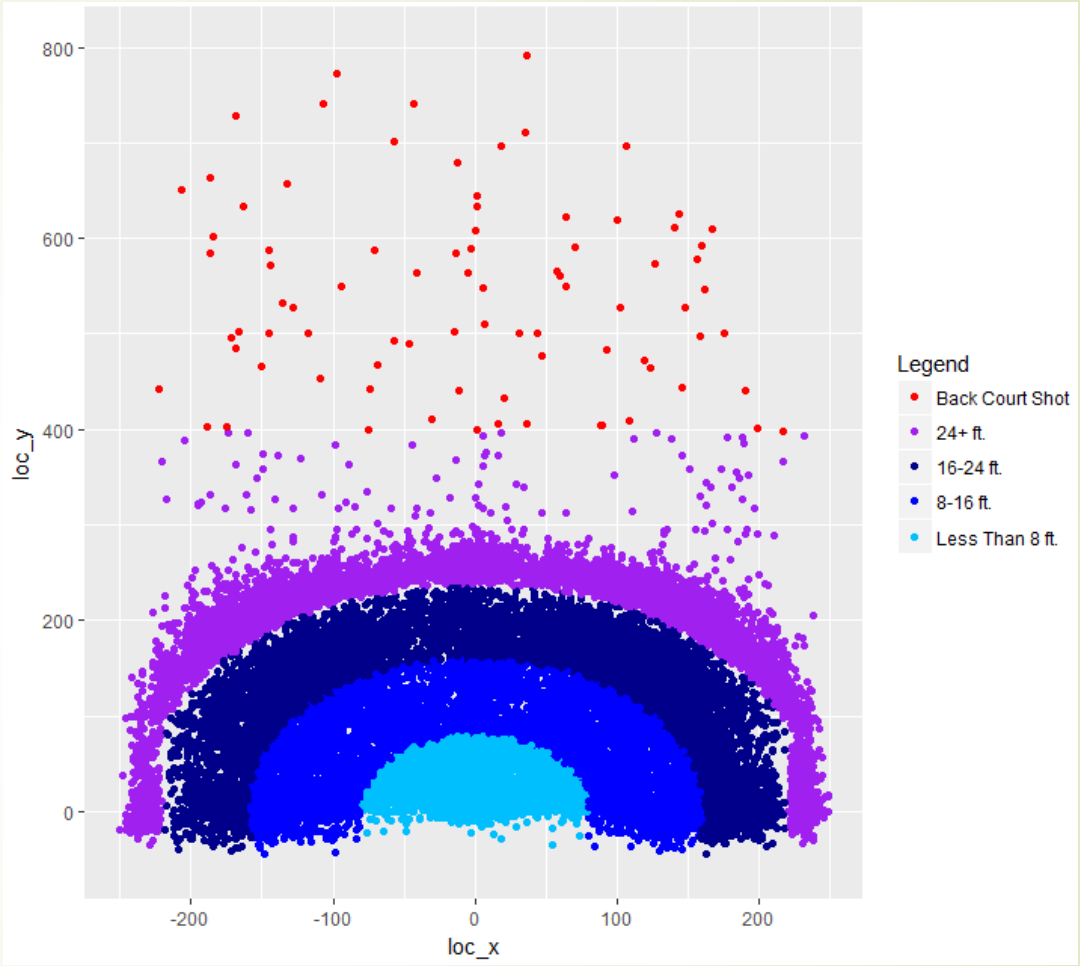
1. Source kobeinit.R to initialize the data frames
2. You can then source kobe_explore.R to generate all plots or select a set to run (ctrl+return). I usually use the x11() device, but you can also use the PDF code.
3. And/or you can run any of the models. Some of the data and plots are sent to files for future reference

Basket Ball Court Dimensions

Scale: 1 ~ 0.1 ft

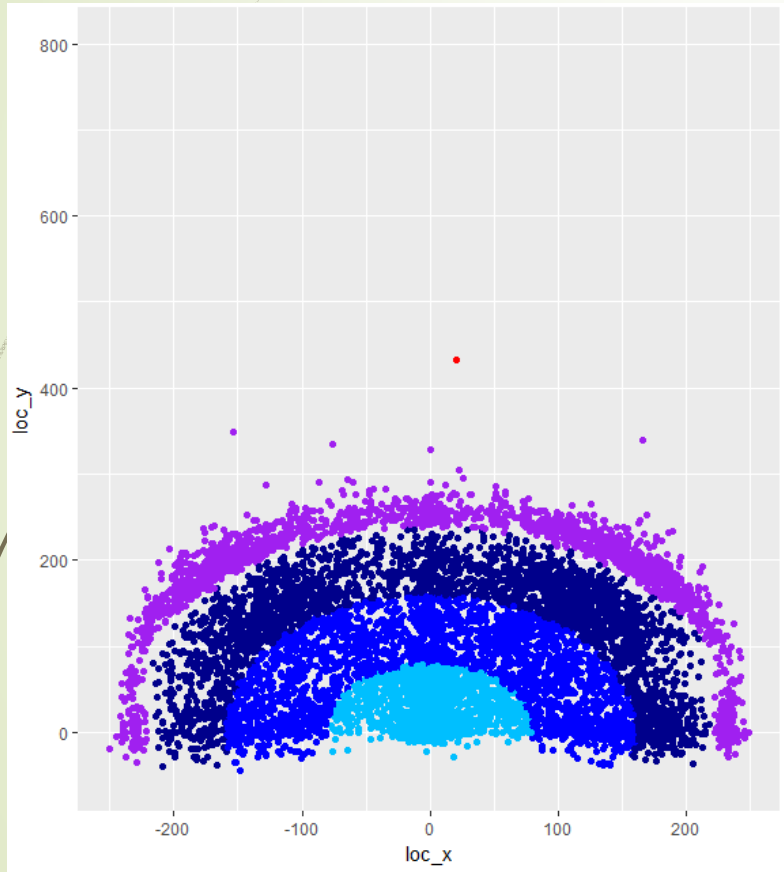


Shot by distance zone
All Shots, including test set

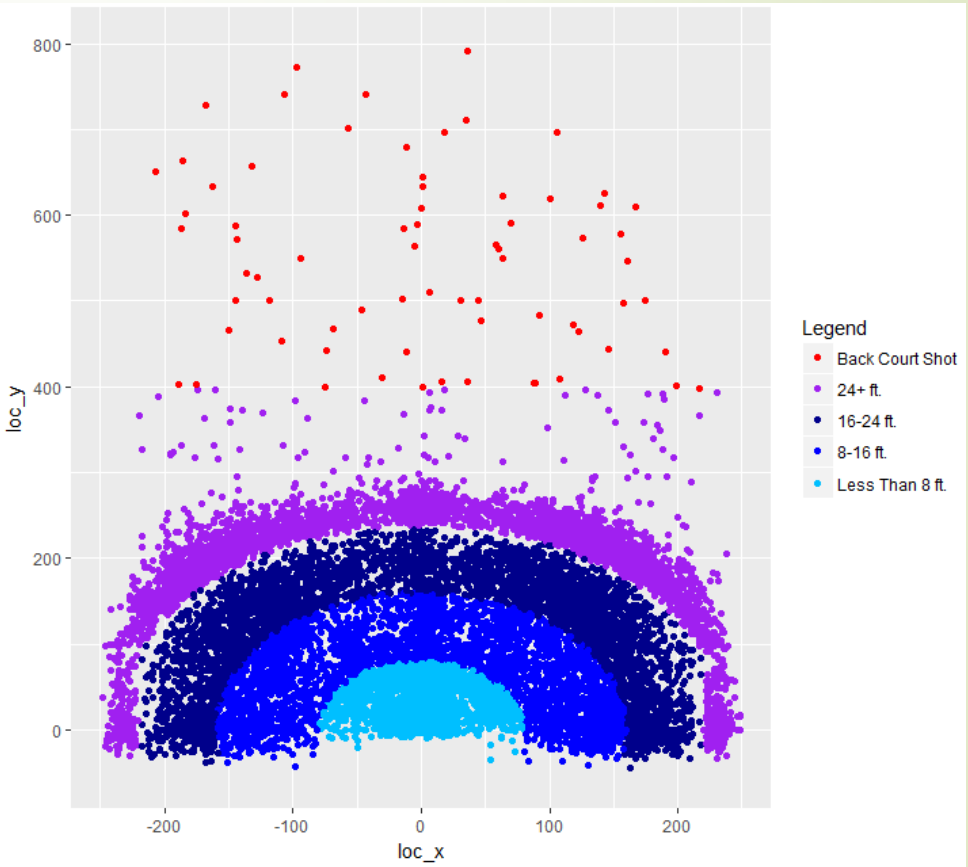


Shot by distance zone made, not made

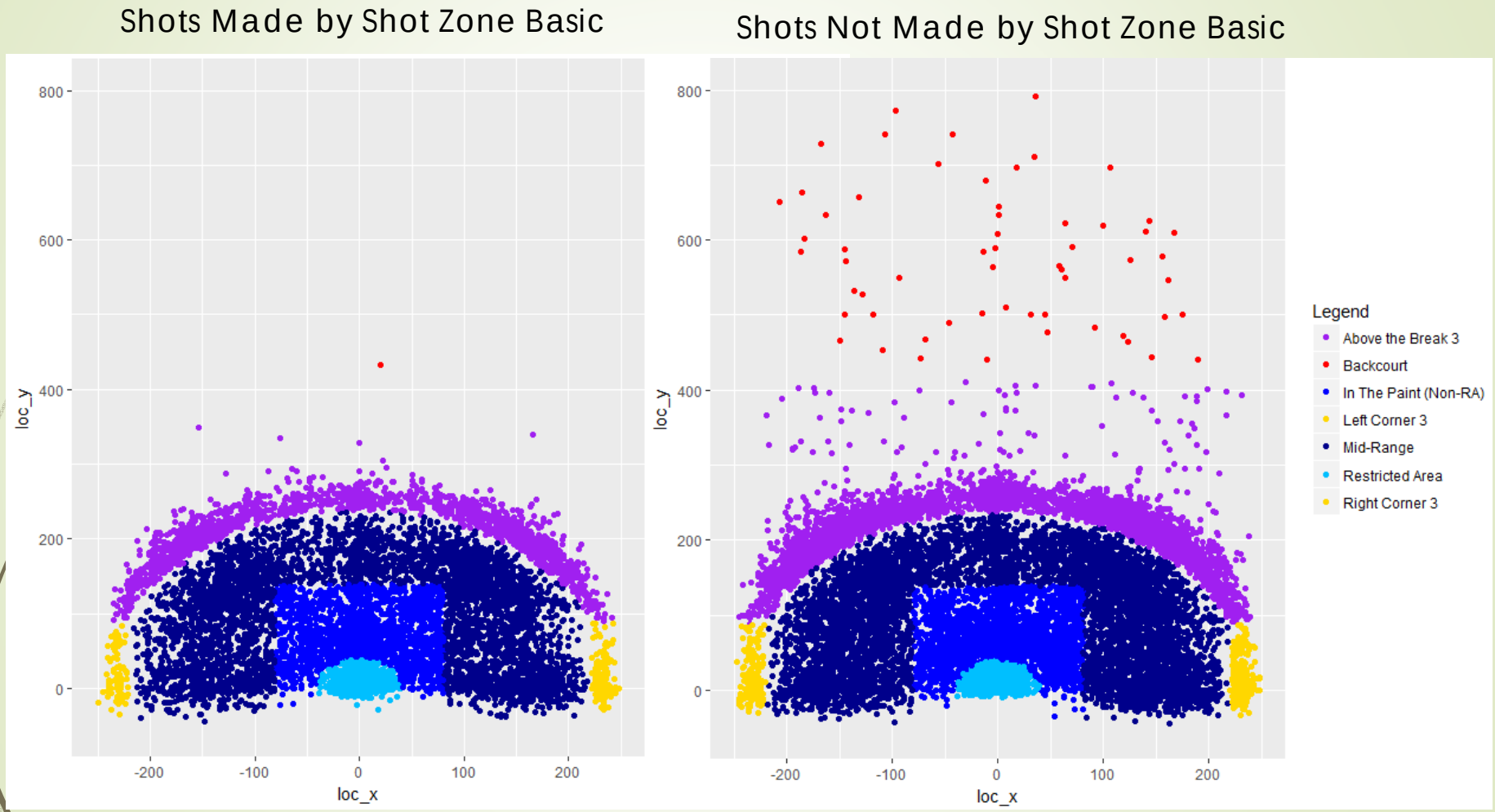
Shots Made



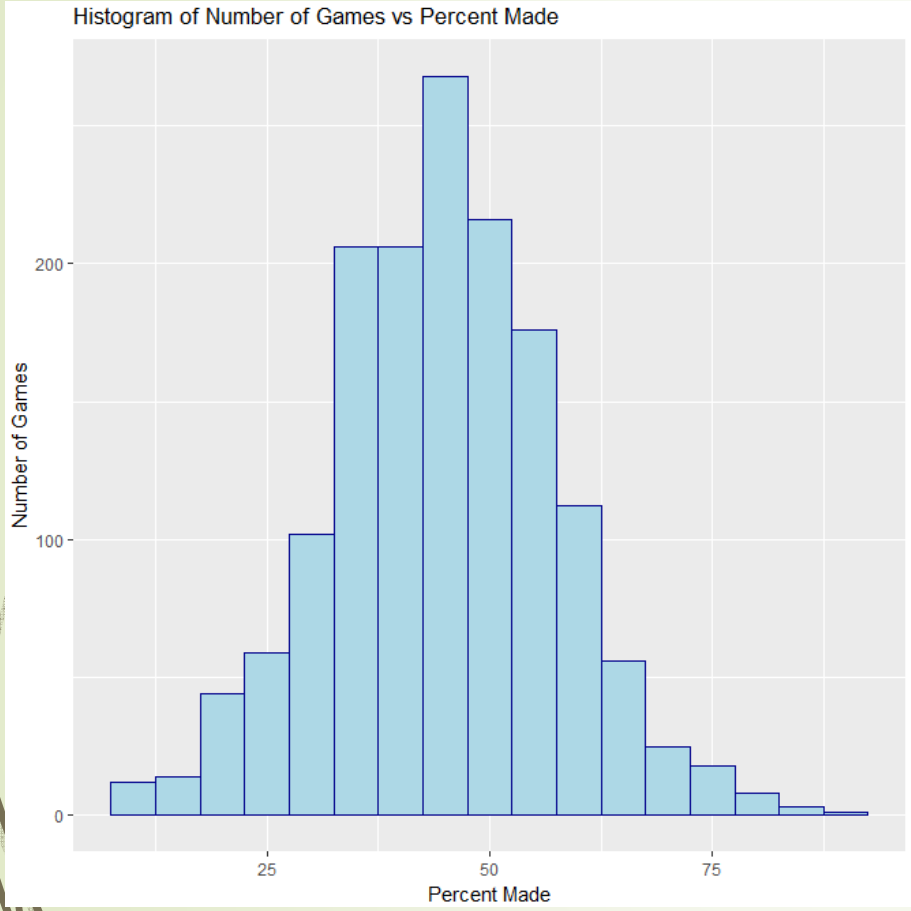
Shots Not Made

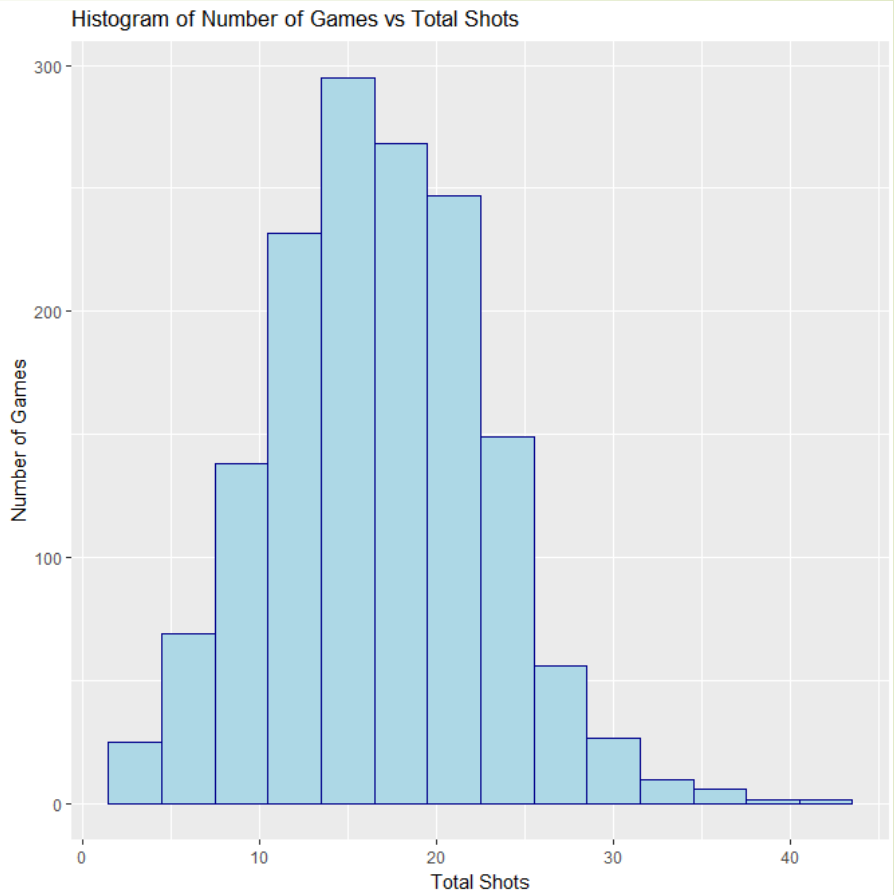


Shots by "shot zone basic"

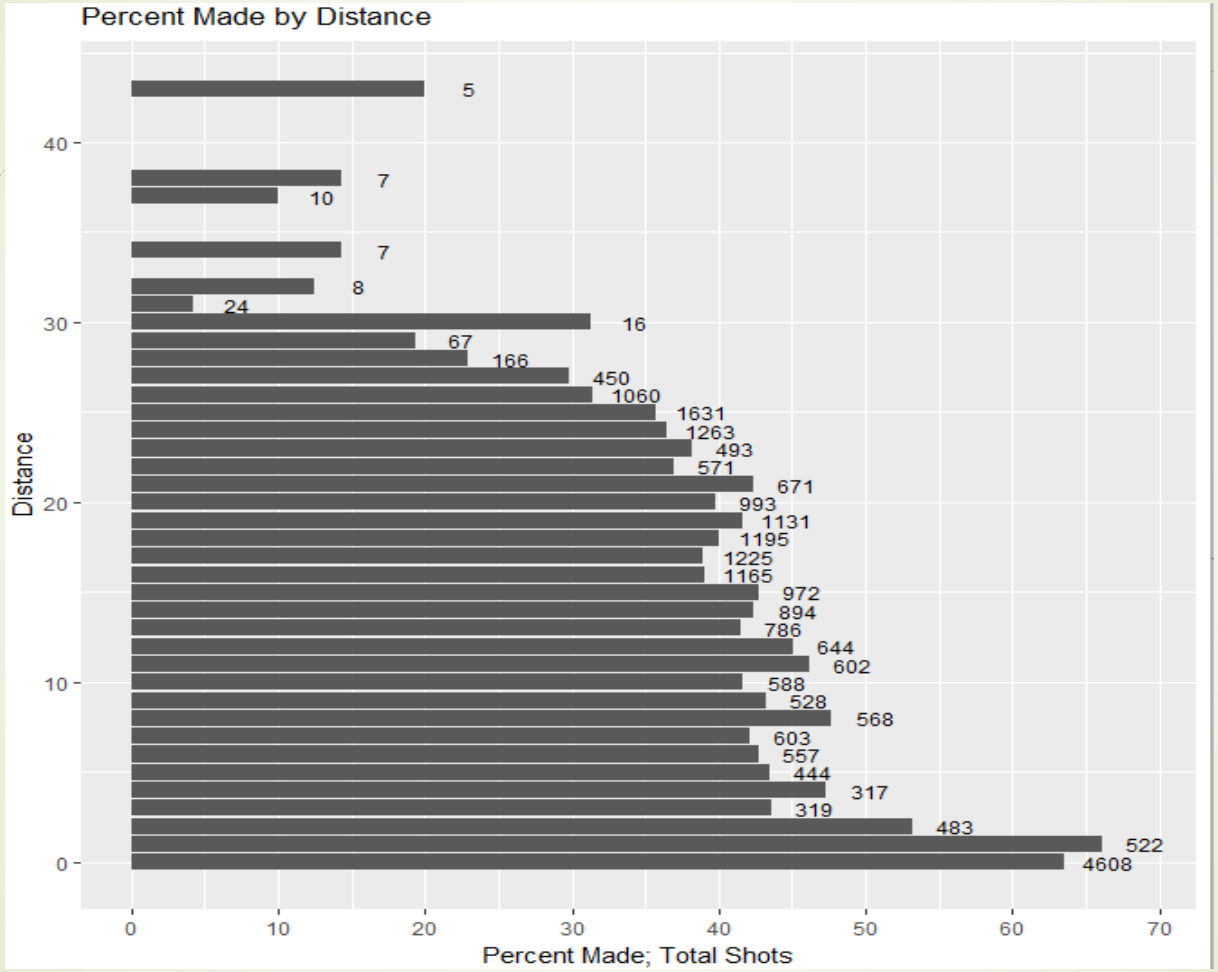


Shots by game

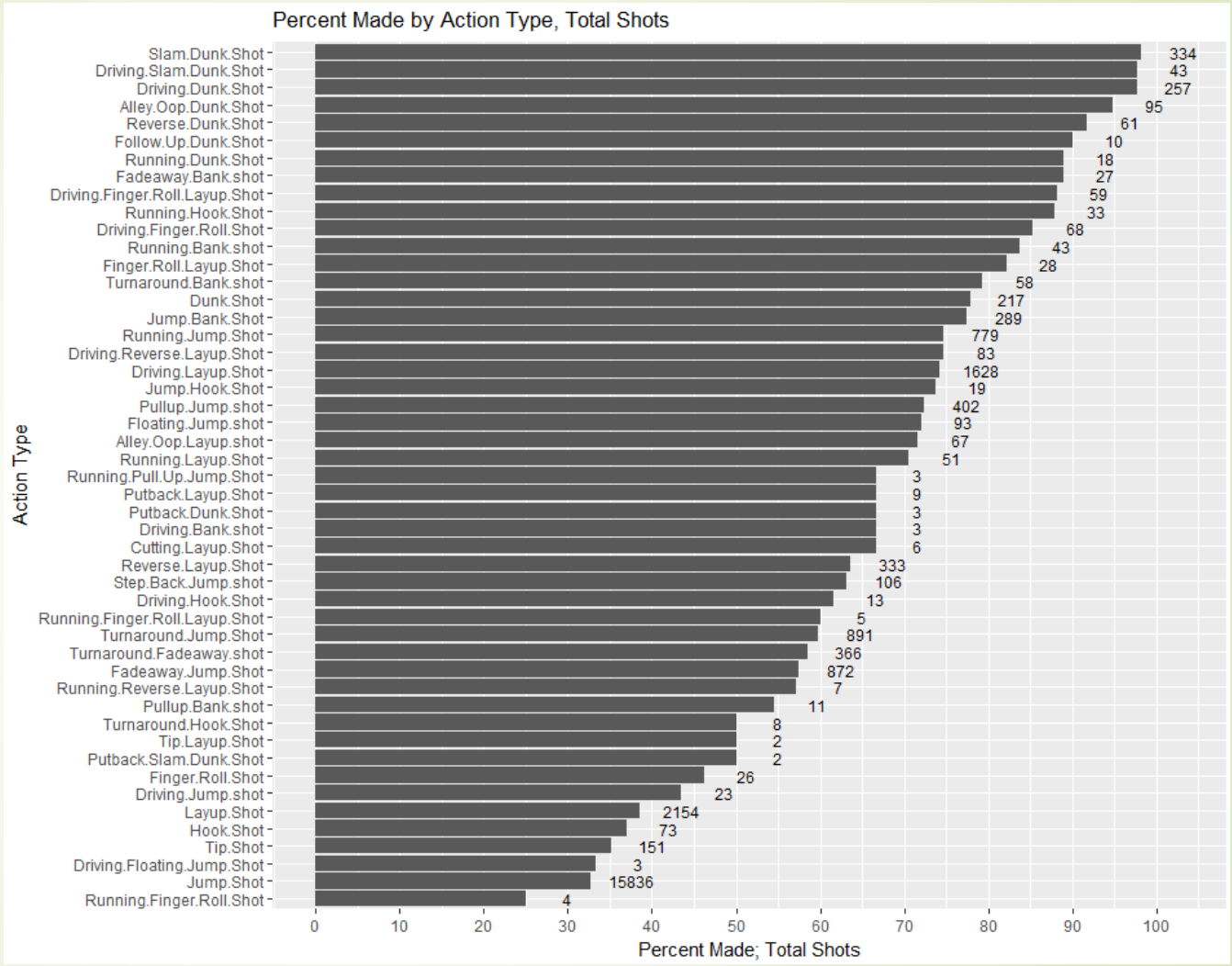




Shots by Distance, Percent Made



Action Type


Percent Made by Action Type, Total Shots
Action Type
Slam.Dunk.Shot 334
Driving.Slam.Dunk.Shot 43
Driving.Dunk.Shot 257
Alley.Oop.Dunk.Shot 95
Reverse.Dunk.Shot 61
Follow.Up.Dunk.Shot 10
Running.Dunk.Shot 18
Fadeaway.Bank.shot 27
Driving.Finger.Roll.Layup.Shot 59
Running.Hook.Shot 33
Driving.Finger.Roll.Shot 68
Running.Bank.shot 43
Finger.Roll.Layup.Shot 28
Turnaround.Bank.shot 58
Dunk.Shot 217
Jump.Bank.Shot 289
Running.Jump.Shot 779
Driving.Reverse.Layup.Shot 83
Driving.Layup.Shot 1628
Jump.Hook.Shot 19
Pullup.Jump.shot 402
Floating.Jump.shot 93
Alley.Oop.Layup.shot 67
Running.Layup.Shot 51
Running.Pull.Up.Jump.Shot 3
Putback.Layup.Shot 9
Putback.Dunk.Shot 3
Driving.Bank.shot 3
Cutting.Layup.Shot 6
Reverse.Layup.Shot 333
Step.Back.Jump.shot 106
Driving.Hook.Shot 13
Running.Finger.Roll.Layup.Shot 5
Turnaround.Jump.Shot 891
Turnaround.Fadeaway.shot 366
Fadeaway.Jump.Shot 872
Running.Reverse.Layup.Shot 7
Pullup.Bank.shot 11
Turnaround.Hook.Shot 8
Tip.Layup.Shot 2
Putback.Slam.Dunk.Shot 2
Finger.Roll.Shot 26
Driving.Jump.shot 23
Layup.Shot 2154
Hook.Shot 73
Tip.Shot 151
Driving.Floating.Jump.Shot 3
Jump.Shot 15836
Running.Finger.Roll.Shot 4
0 10 20 30 40 50 60 70 80 90 100
Percent Made; Total Shots

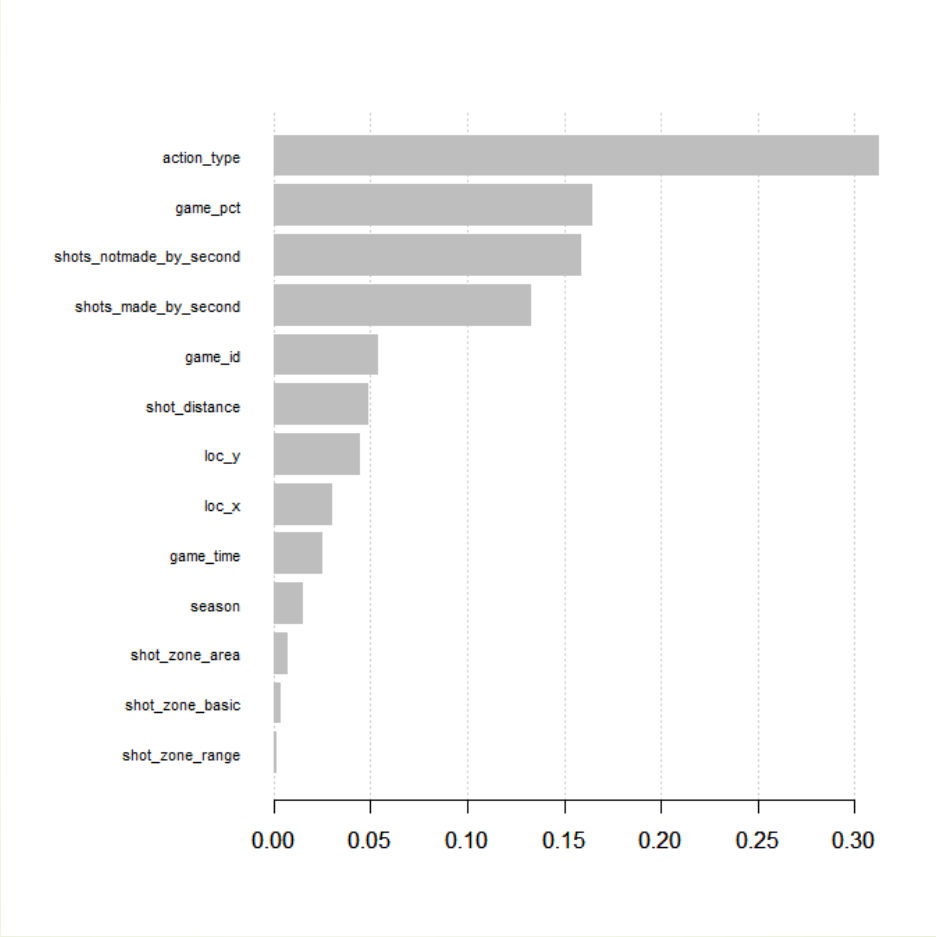
Action_type at 0 distance

	action_type	made
8	Driving.Layup.Shot	1006
17	Layup.Shot	613
31	Slam.Dunk.Shot	309
4	Driving.Dunk.Shot	222
22	Reverse.Layup.Shot	177
11	Dunk.Shot	151
1	Alley.Oop.Dunk.Shot	78
21	Reverse.Dunk.Shot	54
6	Driving.Finger.Roll.Shot	49
32	Tip.Shot	42
2	Alley.Oop.Layup.shot	40
5	Driving.Finger.Roll.Layup.Shot	35
9	Driving.Reverse.Layup.Shot	28
10	Driving.Slam.Dunk.Shot	27
29	Running.Layup.Shot	25
24	Running.Dunk.Shot	14

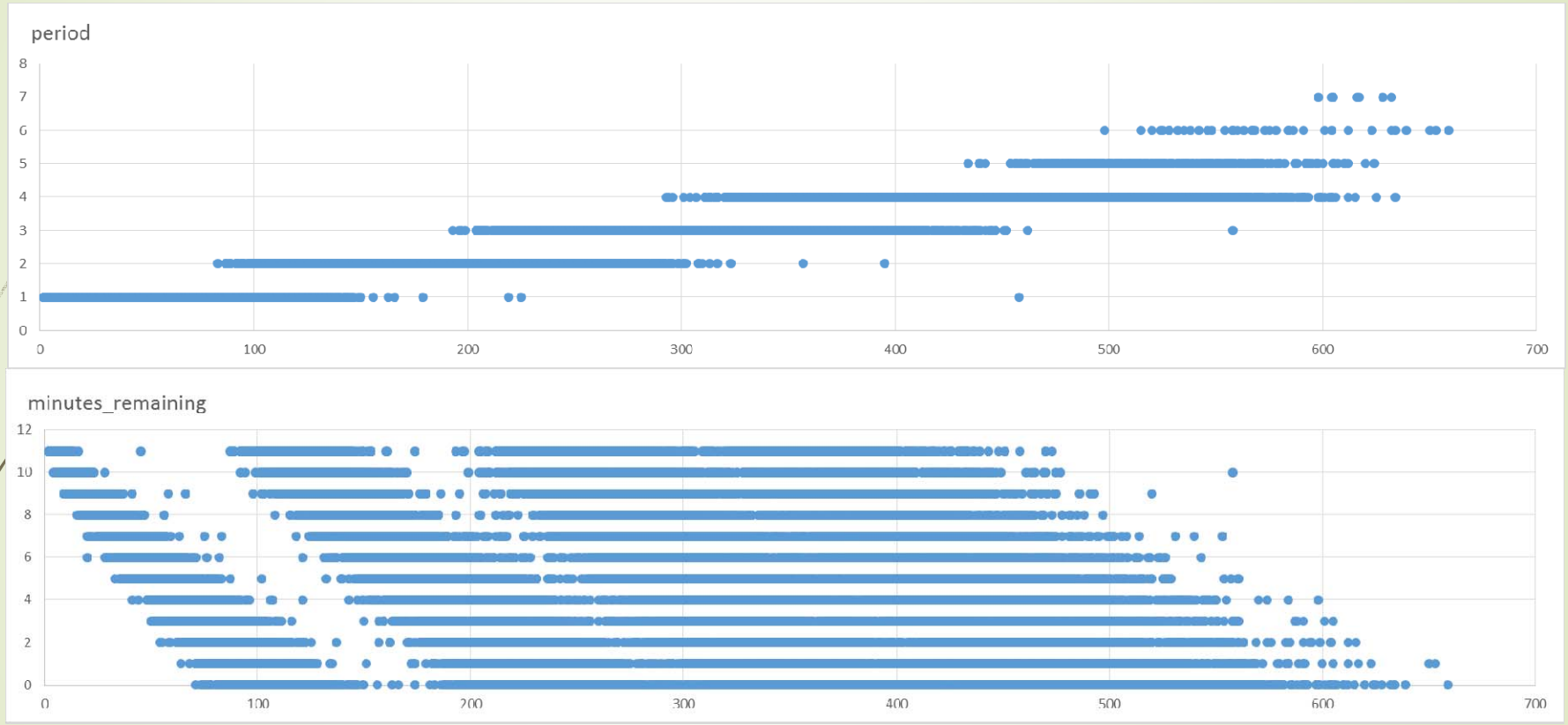
Filter

	action_type	not_made
16	Layup.Shot	1034
7	Driving.Layup.Shot	336
21	Reverse.Layup.Shot	89
30	Tip.Shot	85
10	Dunk.Shot	43
2	Alley.Oop.Layup.shot	13

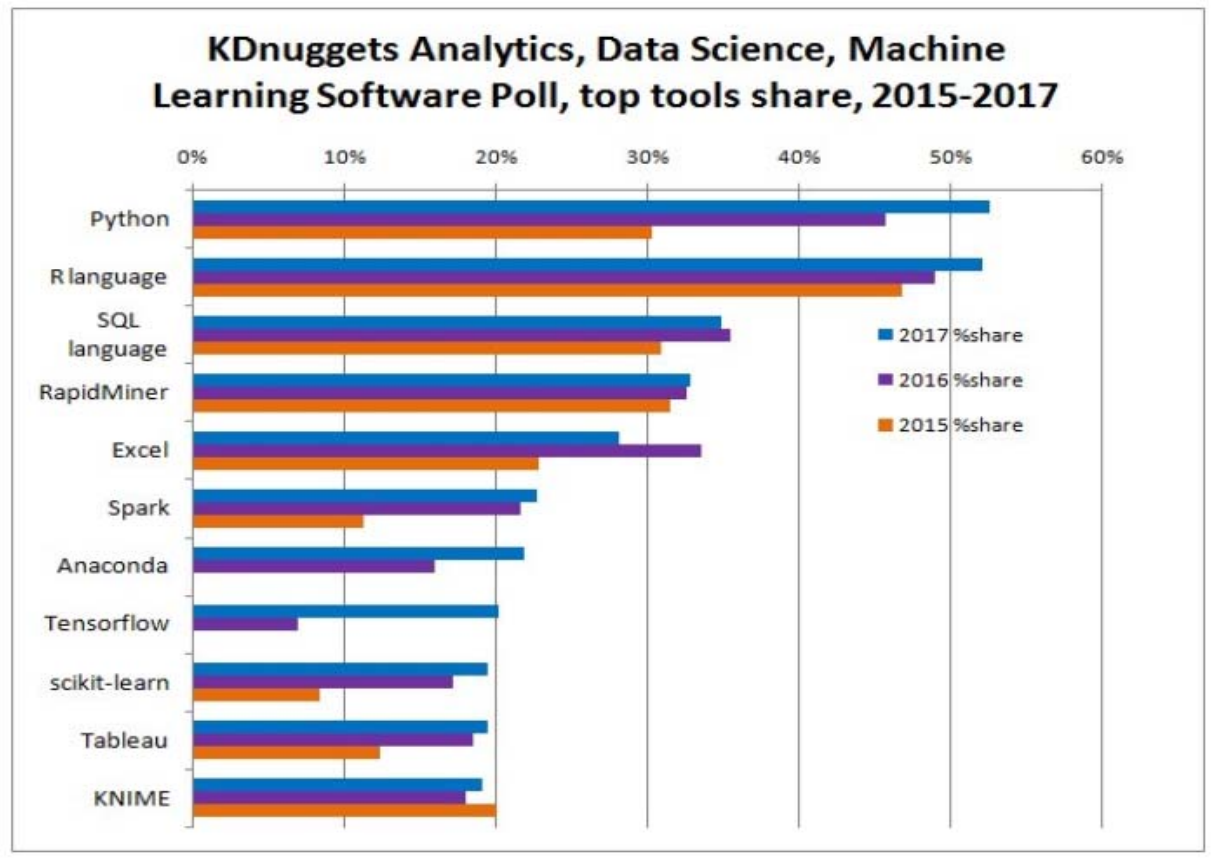
XGBoost importance plot



Game_event_id



Programming environment popularity



https://www.kdnuggets.com/2017/05/poll-analytics-data-science-machine-learning-software-leaders.html