

# INDENG142 Predicting NBA MVP

CASEY LI

12/17/2019

## Data loading and cleaning and splitting

```
# loading data
set.seed(679)

# we cleaned this csv already
csv <- read.csv("NBASStats.csv")
summary(csv)
```

```
##           X           fga           fg3a           fta
## Min.      : 0      Min.      : 3.70      Min.      : 0.000      Min.      : 1.300
## 1st Qu.:162      1st Qu.:13.90      1st Qu.: 0.100      1st Qu.: 4.800
## Median :324      Median :16.80      Median : 1.100      Median : 6.300
## Mean     :324      Mean     :16.53      Mean     : 2.069      Mean     : 6.446
## 3rd Qu.:486      3rd Qu.:19.20      3rd Qu.: 3.700      3rd Qu.: 7.900
## Max.     :648      Max.     :27.80      Max.     :13.200      Max.     :13.100
##
##           per           ts_pct           usg_pct           bpm
## Min.      :10.10      Min.      :0.4410      Min.      : 7.10      Min.      :-2.800
## 1st Qu.:19.90      1st Qu.:0.5430      1st Qu.:23.70      1st Qu.: 2.600
## Median :22.60      Median :0.5680      Median :26.70      Median : 4.500
## Mean     :22.51      Mean     :0.5687      Mean     :26.53      Mean     : 4.601
## 3rd Qu.:24.80      3rd Qu.:0.5960      3rd Qu.:29.90      3rd Qu.: 6.100
## Max.     :31.70      Max.     :0.6990      Max.     :41.70      Max.     :15.600
##
##           vorp           season           player           age
## Min.      :-0.500      1980-81: 31      LeBron James      : 16      Min.      :19.00
## 1st Qu.: 3.100      1981-82: 25      Tim Duncan        : 16      1st Qu.:25.00
## Median : 4.400      1982-83: 23      Karl Malone       : 15      Median :27.00
## Mean     : 4.587      1984-85: 23      Shaquille O'Neal : 14      Mean     :27.47
## 3rd Qu.: 5.800      1998-99: 21      Hakeem Olajuwon  : 13      3rd Qu.:30.00
## Max.     :12.400      1990-91: 20      Kobe Bryant       : 13      Max.     :38.00
##           (Other):506      (Other)           :562
##           win_pct           votes_first           points_won           points_max
## Min.      :0.2195      Min.      : 0.000      Min.      : 1.0      Min.      : 690
## 1st Qu.:0.5610      1st Qu.: 0.000      1st Qu.: 3.0      1st Qu.: 800
## Median :0.6463      Median : 0.000      Median : 23.0      Median :1130
## Mean     :0.6310      Mean     : 6.401      Mean     :166.4      Mean     :1039
## 3rd Qu.:0.7000      3rd Qu.: 1.000      3rd Qu.:184.0      3rd Qu.:1230
## Max.     :0.8902      Max.     :131.000      Max.     :1310.0      Max.     :1310
##
##           award_share           g           mp_per_g           pts_per_g
## Min.      :0.0010      Min.      :17      Min.      :23.60      Min.      : 4.7
## 1st Qu.:0.0040      1st Qu.:73      1st Qu.:34.60      1st Qu.:18.8
## Median :0.0210      Median :79      Median :36.70      Median :22.0
```

```
## Mean :0.1562 Mean :75 Mean :36.33 Mean :22.0
## 3rd Qu.:0.1770 3rd Qu.:81 3rd Qu.:38.30 3rd Qu.:25.7
## Max. :1.0000 Max. :82 Max. :43.70 Max. :37.1
##
## trb_per_g ast_per_g stl_per_g blk_per_g
## Min. : 1.900 Min. : 0.800 Min. :0.200 Min. :0.0000
## 1st Qu.: 4.800 1st Qu.: 2.800 1st Qu.:1.000 1st Qu.:0.3000
## Median : 6.900 Median : 4.300 Median :1.400 Median :0.6000
## Mean : 7.459 Mean : 5.002 Mean :1.428 Mean :0.9817
## 3rd Qu.:10.400 3rd Qu.: 6.700 3rd Qu.:1.800 3rd Qu.:1.4000
## Max. :18.700 Max. :14.500 Max. :3.700 Max. :5.6000
##
## fg_pct fg3_pct ft_pct ws
## Min. :0.3840 Min. :0.0000 Min. :0.4220 Min. : 2.3
## 1st Qu.:0.4630 1st Qu.:0.1670 1st Qu.:0.7370 1st Qu.: 8.5
## Median :0.4920 Median :0.3020 Median :0.7900 Median :10.6
## Mean :0.4944 Mean :0.2576 Mean :0.7802 Mean :10.7
## 3rd Qu.:0.5230 3rd Qu.:0.3650 3rd Qu.:0.8430 3rd Qu.:12.9
## Max. :0.6700 Max. :1.0000 Max. :0.9480 Max. :21.2
##
## ws_per_48
## Min. :0.0460
## 1st Qu.:0.1550
## Median :0.1870
## Mean :0.1877
## 3rd Qu.:0.2180
## Max. :0.3220
##
```

#### #variable selection

```
nba_csv <- select(csv, fg_pct, fg3_pct, ft_pct,
                  trb_per_g, ast_per_g, stl_per_g, blk_per_g,
                  per, ts_pct, usg_pct, ws, bpm, vorp, season,
                  age, award_share, win_pct, pts_per_g)
summary(nba_csv)
```

```
## fg_pct fg3_pct ft_pct trb_per_g
## Min. :0.3840 Min. :0.0000 Min. :0.4220 Min. : 1.900
## 1st Qu.:0.4630 1st Qu.:0.1670 1st Qu.:0.7370 1st Qu.: 4.800
## Median :0.4920 Median :0.3020 Median :0.7900 Median : 6.900
## Mean :0.4944 Mean :0.2576 Mean :0.7802 Mean : 7.459
## 3rd Qu.:0.5230 3rd Qu.:0.3650 3rd Qu.:0.8430 3rd Qu.:10.400
## Max. :0.6700 Max. :1.0000 Max. :0.9480 Max. :18.700
##
## ast_per_g stl_per_g blk_per_g per
## Min. : 0.800 Min. :0.200 Min. :0.0000 Min. :10.10
## 1st Qu.: 2.800 1st Qu.:1.000 1st Qu.:0.3000 1st Qu.:19.90
## Median : 4.300 Median :1.400 Median :0.6000 Median :22.60
## Mean : 5.002 Mean :1.428 Mean :0.9817 Mean :22.51
## 3rd Qu.: 6.700 3rd Qu.:1.800 3rd Qu.:1.4000 3rd Qu.:24.80
## Max. :14.500 Max. :3.700 Max. :5.6000 Max. :31.70
##
## ts_pct usg_pct ws bpm
## Min. :0.4410 Min. : 7.10 Min. : 2.3 Min. : -2.800
```

```
## 1st Qu.:0.5430    1st Qu.:23.70    1st Qu.: 8.5    1st Qu.: 2.600
## Median :0.5680    Median :26.70    Median :10.6    Median : 4.500
## Mean   :0.5687    Mean   :26.53    Mean   :10.7    Mean   : 4.601
## 3rd Qu.:0.5960    3rd Qu.:29.90    3rd Qu.:12.9    3rd Qu.: 6.100
## Max.   :0.6990    Max.   :41.70    Max.   :21.2    Max.   :15.600
##
##      vorp      season      age      award_share
## Min.   :-0.500   1980-81: 31   Min.   :19.00   Min.   :0.0010
## 1st Qu.: 3.100   1981-82: 25   1st Qu.:25.00   1st Qu.:0.0040
## Median : 4.400   1982-83: 23   Median :27.00   Median :0.0210
## Mean   : 4.587   1984-85: 23   Mean   :27.47   Mean   :0.1562
## 3rd Qu.: 5.800   1998-99: 21   3rd Qu.:30.00   3rd Qu.:0.1770
## Max.   :12.400   1990-91: 20   Max.   :38.00   Max.   :1.0000
##
##      (Other):506
##      win_pct      pts_per_g
## Min.   :0.2195   Min.   : 4.7
## 1st Qu.:0.5610   1st Qu.:18.8
## Median :0.6463   Median :22.0
## Mean   :0.6310   Mean   :22.0
## 3rd Qu.:0.7000   3rd Qu.:25.7
## Max.   :0.8902   Max.   :37.1
##
```

```
#separate to training vs. testing set
nba_csv.train <- filter(nba_csv, season != "2018-19")
nba_csv.test <- filter(nba_csv, season == "2018-19")

nba_csv.train <- select(nba_csv.train, -season)
nba_csv.test <- select(nba_csv.test, -season)
```

## Baseline linear model

```
# baseline model -----
baseline <- mean(nba_csv.train$award_share)
pred.base <- rep(baseline, nrow(nba_csv.test))

# OSR2
SSE <- sum((nba_csv.test$award_share - pred.base)^2)
SST = sum((nba_csv.test$award_share - mean(nba_csv.train$award_share))^2)
OSR2 = 1 - SSE/SST
OSR2 #0 as expected! since sse = sst
```

```
## [1] 0
```

## Naive linear regression

```
##linear regression model
testlm <- lm(award_share ~ ., data = nba_csv.train)
summary(testlm)
```

```
##
## Call:
## lm(formula = award_share ~ ., data = nba_csv.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.40274 -0.11786 -0.03753  0.09231  0.69770
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.6529754  0.2288151  -2.854 0.004465 **
## fg_pct       0.5290393  0.3817452   1.386 0.166293
## fg3_pct     -0.0667063  0.0596292  -1.119 0.263708
## ft_pct       0.1576860  0.1352366   1.166 0.244062
## trb_per_g    0.0071437  0.0039831   1.793 0.073381 .
## ast_per_g    0.0163803  0.0046416   3.529 0.000448 ***
## stl_per_g   -0.0447344  0.0186352  -2.401 0.016665 *
## blk_per_g    0.0045502  0.0122848   0.370 0.711213
## per          0.0005473  0.0092639   0.059 0.952907
## ts_pct     -1.3495734  0.4553094  -2.964 0.003152 **
## usg_pct      0.0038880  0.0058959   0.659 0.509854
## ws           0.0237721  0.0097291   2.443 0.014828 *
## bpm          0.0140661  0.0164185   0.857 0.391929
## vorp         0.0085559  0.0244173   0.350 0.726154
## age          0.0018868  0.0020919   0.902 0.367426
## win_pct      0.6003816  0.0898235   6.684 5.19e-11 ***
## pts_per_g    0.0104445  0.0044217   2.362 0.018479 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.1783 on 620 degrees of freedom
## Multiple R-squared:  0.5303, Adjusted R-squared:  0.5181
## F-statistic: 43.74 on 16 and 620 DF, p-value: < 2.2e-16
```

```
testlm$coefficients
```

```
##      (Intercept)      fg_pct      fg3_pct      ft_pct      trb_per_g
## -0.6529753819  0.5290393304 -0.0667062644  0.1576860037  0.0071437321
##      ast_per_g      stl_per_g      blk_per_g      per      ts_pct
##  0.0163803226 -0.0447344375  0.0045502399  0.0005473181 -1.3495734159
##      usg_pct      ws      bpm      vorp      age
##  0.0038880487  0.0237721109  0.0140660859  0.0085559299  0.0018868444
##      win_pct      pts_per_g
##  0.6003815569  0.0104444661
```

```
testlm$fitted.values
```

```
##           1           2           3           4           5
## 0.3972906118 0.2916581793 0.3594884314 0.1993401127 0.1624579346
##           6           7           8           9          10
## 0.2147955046 0.2154722659 0.0350259089 -0.0589241505 0.0297343558
##          11          12          13          14          15
## 0.0885785342 0.0922945096 -0.2112985691 -0.0617071877 -0.2187128273
```

##	16	17	18	19	20
##	-0.0637384935	-0.0691877979	-0.0106819405	0.0264315739	-0.0688684067
##	21	22	23	24	25
##	-0.0610597527	0.0219044207	-0.0774595865	-0.2117189962	-0.0560655893
##	26	27	28	29	30
##	-0.2113985352	0.0414616602	-0.2139183151	-0.0582458470	-0.1094189856
##	31	32	33	34	35
##	-0.1195729109	0.3798380306	0.3958375379	0.3154756571	0.2087001326
##	36	37	38	39	40
##	0.1747815062	0.1971712924	0.1797696935	0.2800780058	0.1998916974
##	41	42	43	44	45
##	0.2261494109	-0.0338753795	-0.1361760292	0.1479482157	-0.0056570763
##	46	47	48	49	50
##	0.0354081373	0.0028759112	-0.1874816730	-0.0294870068	-0.0377728547
##	51	52	53	54	55
##	-0.2869335512	-0.0299859784	0.0449853754	-0.2319481805	-0.0200816963
##	56	57	58	59	60
##	-0.1276560415	0.3810857245	0.3798848301	0.2773002294	0.1782551794
##	61	62	63	64	65
##	0.2660436783	0.2175688987	0.0121898546	0.0634733313	0.1016404280
##	66	67	68	69	70
##	0.1816038097	0.0680002057	0.0099041576	-0.1061899541	-0.0208888189
##	71	72	73	74	75
##	-0.2615872467	-0.0231199347	-0.0724331019	-0.0484033035	0.0598843630
##	76	77	78	79	80
##	0.0001164818	0.1760449005	-0.0945440380	0.0422575085	0.4607651897
##	81	82	83	84	85
##	0.1277977964	0.1692539986	0.0933294478	0.1330411785	0.1622708841
##	86	87	88	89	90
##	0.2235146215	0.1337276646	-0.0199162344	0.0740147508	0.1296931373
##	91	92	93	94	95
##	0.0497300998	-0.1157686670	-0.2275910100	0.0754248231	0.5651062663
##	96	97	98	99	100
##	0.2964140129	0.2152980481	0.2495141359	0.2393991395	0.2414150315
##	101	102	103	104	105
##	-0.0218312096	0.1908948211	0.2385215484	-0.0028557656	0.0988968475
##	106	107	108	109	110
##	0.1689760146	0.0692480212	0.0512158342	-0.1016887179	-0.1684487839
##	111	112	113	114	115
##	0.0833802641	-0.0704482056	-0.0474338351	-0.1014580300	-0.2061901511
##	116	117	118	119	120
##	-0.0467824444	0.0588340294	0.5485016823	0.2255040542	0.3133382257
##	121	122	123	124	125
##	0.1269333208	0.2673610796	0.1467539149	0.1221774615	0.1766752978
##	126	127	128	129	130
##	0.0760933393	0.0616564975	0.0539584230	0.1066611894	0.2511496383
##	131	132	133	134	135
##	0.0830623377	-0.1425847923	0.1916543275	-0.1859185553	0.0629914075
##	136	137	138	139	140
##	0.5505936853	0.4431377734	0.4963458117	0.3543969927	0.3164025261
##	141	142	143	144	145
##	0.1851487704	0.1425950248	0.1045610563	0.0627110386	0.0749202261
##	146	147	148	149	150
##	-0.2070096665	-0.0476501164	0.0902816348	0.0450707472	0.1280898977

##	151	152	153	154	155
##	-0.0058145069	-0.1013492495	0.6435355211	0.4909543444	0.3038973946
##	156	157	158	159	160
##	0.2664241078	0.3199673969	0.2150042519	0.1241270757	0.1435291862
##	161	162	163	164	165
##	0.2010242045	0.0534988670	0.1527539107	0.0779074322	-0.0970659812
##	166	167	168	169	170
##	-0.1393687167	0.1538468443	-0.0045354729	-0.3759730112	0.4824391599
##	171	172	173	174	175
##	0.6038916276	0.3256121344	0.1378877341	0.1852858100	0.3272939319
##	176	177	178	179	180
##	0.1846989439	0.2253351013	0.1340769873	0.0928675085	0.0644146218
##	181	182	183	184	185
##	0.0382743230	-0.1410676510	0.0811748639	0.0874044658	0.1056544324
##	186	187	188	189	190
##	0.0339469554	-0.0037865435	0.1074438670	0.4979090700	0.3760578386
##	191	192	193	194	195
##	0.5844865604	0.4009488401	0.2607127947	0.3299225644	0.1882687979
##	196	197	198	199	200
##	0.1648916237	0.2037271630	0.2957021013	-0.0459284286	0.3085042573
##	201	202	203	204	205
##	0.0431050342	0.1212627625	0.6359032968	0.4140381274	0.4313950629
##	206	207	208	209	210
##	0.2695938546	0.4029921390	0.3203995910	0.2290042195	0.1570796380
##	211	212	213	214	215
##	0.1244123724	0.1554239678	0.1603209640	0.1928117616	-0.0806575671
##	216	217	218	219	220
##	-0.0144751477	0.0526305755	-0.1117966994	-0.0630058446	0.1470418841
##	221	222	223	224	225
##	0.0808047232	0.0105841321	0.6091494637	0.3648177724	0.2684030535
##	226	227	228	229	230
##	0.3626741179	0.2475786360	0.1182449365	0.0379832498	0.1550542683
##	231	232	233	234	235
##	0.3797606469	0.0515001160	0.2042012694	0.0926412763	0.1744586967
##	236	237	238	239	240
##	0.1097527419	0.1308838425	0.0474664250	-0.0565397787	0.4397763005
##	241	242	243	244	245
##	0.4851069388	0.5549538297	0.2523358110	0.1472741115	0.2774255022
##	246	247	248	249	250
##	0.1013358223	0.3397837546	0.0501789429	0.1697483450	0.0582189030
##	251	252	253	254	255
##	-0.0532288389	0.0597979040	0.0058044398	0.4376235050	0.6531149024
##	256	257	258	259	260
##	0.2310999456	0.4367468762	0.3248808135	0.0638498489	0.1999928553
##	261	262	263	264	265
##	0.3433020874	0.0330149448	0.1642114058	0.1435191340	0.0545589017
##	266	267	268	269	270
##	-0.0724093291	-0.0560118081	0.1413972241	0.0131052507	0.1031591960
##	271	272	273	274	275
##	0.5127593147	0.4017993440	0.3695351133	0.1846791855	0.2434262271
##	276	277	278	279	280
##	0.2427020446	0.1452753090	0.1630967564	0.1498564072	0.1166625696
##	281	282	283	284	285
##	-0.0244534565	-0.0783888995	-0.2711467691	0.0844513004	-0.1519106289

##	286	287	288	289	290
##	0.6754691688	0.5265069310	0.2726951859	0.2123914072	0.3346789889
##	291	292	293	294	295
##	0.1839569039	0.3854044001	0.1421588108	0.2027522146	0.1720069621
##	296	297	298	299	300
##	0.0964786602	0.1018918351	-0.1529203937	0.0557838077	0.0356600415
##	301	302	303	304	305
##	0.1298851865	-0.1073350671	0.5759807021	0.5925164560	0.3723969641
##	306	307	308	309	310
##	0.2886361909	0.0527791256	0.2488176198	0.1721828614	0.1725231196
##	311	312	313	314	315
##	0.1416229339	0.2583806222	0.3367462913	0.0996400901	0.0734099288
##	316	317	318	319	320
##	-0.0115412307	0.1420356117	0.1726953571	-0.0631088225	-0.2676841750
##	321	322	323	324	325
##	0.0075989455	0.0310160099	0.4547717182	0.4937551757	0.2388624895
##	326	327	328	329	330
##	0.3182775521	0.2889096306	0.2026247901	0.3406526970	0.1043198074
##	331	332	333	334	335
##	0.0946830229	0.1073807719	-0.0242678095	-0.1425502496	0.0218473476
##	336	337	338	339	340
##	-0.0454342904	-0.1795787294	0.0589440775	0.0112058018	-0.2012647130
##	341	342	343	344	345
##	0.0473286205	0.2827174908	0.1104186216	0.2056320401	0.0488571816
##	346	347	348	349	350
##	0.0426866942	0.2113561972	-0.0543326756	0.1021685191	0.0312985168
##	351	352	353	354	355
##	0.0116338360	-0.1305942693	0.1261982554	-0.0400401604	-0.0381475952
##	356	357	358	359	360
##	-0.0809917541	-0.1838096964	-0.0638212588	-0.1662626945	-0.1539933938
##	361	362	363	364	365
##	-0.0992972190	-0.2000240707	0.7518793147	0.2629045110	0.2310265738
##	366	367	368	369	370
##	0.4207412507	0.3261598385	0.3123144386	0.0748923396	0.1398683916
##	371	372	373	374	375
##	0.2089471384	0.1706762766	0.0978933305	0.2717219537	-0.0849083885
##	376	377	378	379	380
##	0.0120082629	-0.0231967310	-0.0044980131	0.2955677828	0.3409694031
##	381	382	383	384	385
##	0.5101685406	0.3543101636	0.2438922165	0.2347969567	0.2986949701
##	386	387	388	389	390
##	0.1347352462	0.2640186099	0.1663920229	0.1560350422	0.2552806599
##	391	392	393	394	395
##	-0.0268584633	0.0771730730	0.0701336769	0.0840772540	-0.0198988233
##	396	397	398	399	400
##	0.5541818708	0.1434521102	0.4151199967	0.2114993287	0.3078507897
##	401	402	403	404	405
##	0.2525881053	0.2716862758	0.2075810494	0.1058123254	0.0024866901
##	406	407	408	409	410
##	0.1606126894	0.3097576893	-0.0216198066	0.0712699715	-0.0117237312
##	411	412	413	414	415
##	-0.0786940550	0.1141863768	0.0598296074	0.5055873820	0.4545650523
##	416	417	418	419	420
##	0.3738324722	0.4242792031	0.3417190884	0.1192522821	0.3656799833

##	421	422	423	424	425
##	0.0009865580	0.2044291653	0.2205610144	0.0971571942	0.1655458362
##	426	427	428	429	430
##	0.1252376917	0.6222301662	0.3821802498	0.2055959013	0.1208260350
##	431	432	433	434	435
##	0.2049859192	0.2311936860	0.0726137876	0.0600129005	-0.1193174654
##	436	437	438	439	440
##	0.2388294366	0.0393077360	0.1275300220	0.0395334939	-0.1248931990
##	441	442	443	444	445
##	0.0299705095	-0.0375466543	0.1412967430	0.3028676155	0.3671230655
##	446	447	448	449	450
##	0.2885955517	0.1717485048	0.3103513373	0.2939121416	0.2499642446
##	451	452	453	454	455
##	0.2853173546	0.1313096556	0.4147423914	0.0875501986	0.0876352596
##	456	457	458	459	460
##	-0.3260388892	-0.0826642721	0.2002130318	0.1826344976	0.4978785172
##	461	462	463	464	465
##	0.4568226373	0.3912972646	0.2858952458	0.3706454185	0.2931539653
##	466	467	468	469	470
##	0.2609887046	0.1242843550	0.1879032142	0.2104383710	0.4650738959
##	471	472	473	474	475
##	0.2266249078	0.2477688589	0.3297937109	0.3418850003	0.2313973623
##	476	477	478	479	480
##	0.0572553288	0.1131500422	0.1440386829	0.1057884734	0.0664411684
##	481	482	483	484	485
##	-0.1766805599	0.1746594614	0.0880567587	0.0545105124	-0.0569501919
##	486	487	488	489	490
##	0.0858021939	0.3343997339	0.4482708251	0.3385180182	0.4643158179
##	491	492	493	494	495
##	0.1245032474	0.2273514190	0.2536885914	0.0945787154	0.0689771121
##	496	497	498	499	500
##	0.1394859743	0.2397800795	0.1199733949	0.0403149811	0.1384325136
##	501	502	503	504	505
##	-0.0455375356	0.1996603580	-0.0186272146	0.7479513246	0.3592312891
##	506	507	508	509	510
##	0.4192197380	0.2587521795	0.4537657841	-0.0977996549	0.1078486447
##	511	512	513	514	515
##	0.1492435669	0.2493614994	0.1430246011	0.2232322124	0.0470581456
##	516	517	518	519	520
##	0.6828542433	0.3022433780	0.2171480860	0.2219942794	0.3411190131
##	521	522	523	524	525
##	0.1017282697	0.2068840203	0.1041113051	0.1226431847	0.0581861609
##	526	527	528	529	530
##	0.0194981166	-0.0037236448	0.0173314272	-0.1620522282	0.0644666630
##	531	532	533	534	535
##	0.4071098911	0.2350743535	0.4642265111	0.2401642703	0.1832273014
##	536	537	538	539	540
##	0.1713661222	0.3041568912	0.0994611354	-0.0110512405	0.0099821954
##	541	542	543	544	545
##	-0.0298439333	0.0596330567	0.1250847756	0.4436245018	0.2597127397
##	546	547	548	549	550
##	0.1784332673	0.1156220949	0.1152796829	0.0206909056	0.0145201226
##	551	552	553	554	555
##	-0.0789239465	-0.2026512648	0.1446220773	0.1722155800	-0.0369104644



```
##          556          557          558          559          560
## 0.1352739503 0.0630749572 -0.0768576577 0.6351406025 0.4657458941
##          561          562          563          564          565
## 0.1762673642 0.2195649413 0.1946656498 0.1448003329 0.1529103111
##          566          567          568          569          570
## 0.1295497713 0.3018088651 0.1910605142 0.1063599355 -0.1565167091
##          571          572          573          574          575
## 0.1705731941 -0.0396804970 0.0655790197 -0.0342479090 0.5666388623
##          576          577          578          579          580
## 0.3876921964 0.2508491256 0.1510339862 0.1876706611 0.2579452923
##          581          582          583          584          585
## 0.2307947276 0.0258071992 0.1224271653 0.1211974817 0.2833279841
##          586          587          588          589          590
## 0.1357306606 -0.0068033097 0.0667963046 0.0861510717 0.0052273728
##          591          592          593          594          595
## -0.0208128934 0.4224504356 0.4031673460 0.2609821527 0.3697239643
##          596          597          598          599          600
## 0.2391993075 0.3483503599 0.0877582441 0.1518206918 0.1714648424
##          601          602          603          604          605
## 0.1069709819 -0.0232385474 0.0816534859 0.6079772493 0.2741201940
##          606          607          608          609          610
## 0.4211225832 0.4673510961 0.3687107001 0.2637359703 0.2303198017
##          611          612          613          614          615
## 0.0986823029 0.2387959324 0.1662542860 0.6544219717 0.5206176929
##          616          617          618          619          620
## 0.2988363183 0.3648256025 0.2448380467 0.3206963365 0.1791531761
##          621          622          623          624          625
## 0.1629478848 0.0819950458 0.3325198717 0.1157292559 0.5587262244
##          626          627          628          629          630
## 0.4365125411 0.2365969130 0.2585384265 0.3725967368 0.2019674734
##          631          632          633          634          635
## 0.2344634968 0.1412535188 0.1384792308 -0.0036472864 0.1466540818
##          636          637
## 0.0507897436 -0.0023127817
```

```
## testing multicollinearity
vif(testlm)
```

```
##      fg_pct  fg3_pct  ft_pct trb_per_g ast_per_g stl_per_g blk_per_g
## 5.920908  1.690224  2.797433  3.440902  3.472220  2.389805  2.630610
##      per  ts_pct  usg_pct      ws      bpm      vorp      age
## 23.899302  6.350981 16.487403 21.594223 39.249989 53.801023 1.215793
##      win_pct pts_per_g
## 1.884890 10.375766
```

```
## predicting
predictions_testlm <- predict(testlm, newdata=nba_csv.test)
predictions_testlm
```

```
##          1          2          3          4          5          6          7
## 0.4918244 0.5519403 0.1647240 0.3149604 0.1654079 0.2545112 0.1870386
##          8          9         10         11         12
## 0.2452853 0.1413281 0.2404894 0.1355675 0.1319142
```

```
nba_csv.test
```

```
##      fg_pct fg3_pct ft_pct trb_per_g ast_per_g stl_per_g blk_per_g per
## 1  0.578  0.256  0.729    12.5      5.9      1.3      1.5 30.9
## 2  0.442  0.368  0.879     6.6      7.5      2.0      0.7 30.6
## 3  0.438  0.386  0.839     8.2      4.1      2.2      0.4 23.3
## 4  0.511  0.307  0.821    10.8      7.3      1.4      0.7 26.3
## 5  0.472  0.437  0.916     5.3      5.2      1.3      0.4 24.4
## 6  0.444  0.369  0.912     4.6      6.9      1.1      0.4 23.7
## 7  0.484  0.300  0.804    13.6      3.7      0.7      1.9 26.1
## 8  0.521  0.353  0.885     6.4      5.9      0.7      1.1 24.2
## 9  0.496  0.371  0.854     7.3      3.3      1.8      0.4 25.8
## 10 0.428  0.290  0.656    11.1     10.7      1.9      0.5 21.1
## 11 0.669  0.000  0.636    12.9      2.0      0.8      2.3 24.6
## 12 0.510  0.339  0.665     8.5      8.3      1.3      0.6 25.6
##      ts_pct usg_pct   ws  bpm  vorp age award_share  win_pct pts_per_g
## 1  0.644   32.3 14.4 10.8  7.6  24      0.932 0.7317073      27.7
## 2  0.616   40.5 15.2 11.7  9.9  29      0.768 0.6463415      36.1
## 3  0.583   29.5 11.9  5.5  5.3  28      0.352 0.5975610      28.0
## 4  0.589   27.4 11.8  9.5  7.3  23      0.210 0.6585366      20.1
## 5  0.641   30.4  9.7  6.3  4.9  30      0.173 0.6951220      27.3
## 6  0.588   29.3 12.1  5.5  5.4  28      0.068 0.6463415      25.8
## 7  0.593   33.3  8.7  4.1  3.3  24      0.049 0.6219512      27.5
## 8  0.631   29.0 11.5  4.3  4.3  30      0.025 0.6951220      26.0
## 9  0.606   30.3  9.5  5.0  3.6  27      0.013 0.7073171      26.6
## 10 0.501   30.9  6.8  6.5  5.6  30      0.008 0.5975610      22.9
## 11 0.682   17.8 14.4  7.0  5.9  26      0.001 0.6097561      15.9
## 12 0.588   31.6  7.2  8.1  4.9  34      0.001 0.4512195      27.4
```

```
SSE <- sum((nba_csv.test$award_share - predictions_testlm)^2)
SST = sum((nba_csv.test$award_share - mean(nba_csv.train$award_share))^2)
OSR2 = 1 - SSE/SST
OSR2 #0.5684896
```

```
## [1] 0.5684896
```

## Backwards stepwise linear regression

```
# backwards stepwise regression -----
set.seed(679)

# training model
step.model <- train(award_share ~., data = nba_csv.train,
  method = "leapBackward",
  tuneGrid = data.frame(nvmax = 1:17),
  trControl = trainControl(method = "cv", number = 5)
)
step.model$results
```

```
##      nvmax      RMSE Rsquared      MAE      RMSESD RsquaredSD      MAESD
```

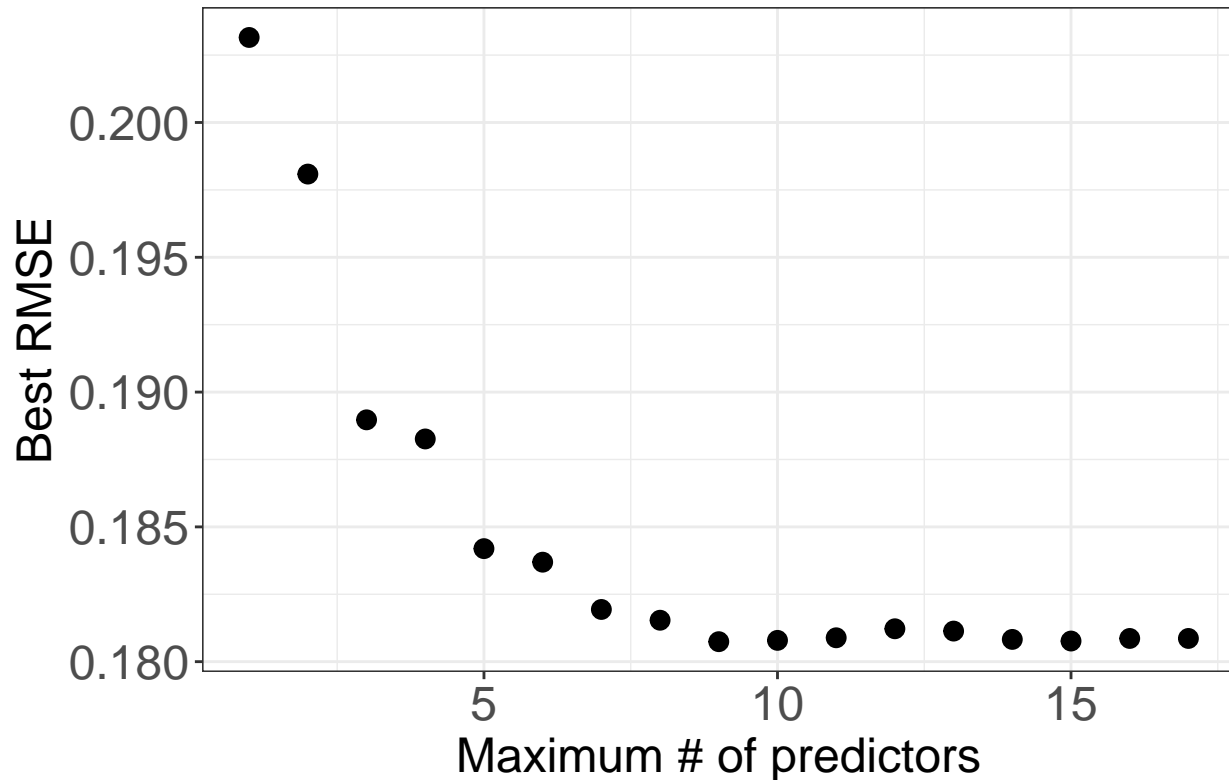
```
## 1      1 0.2031545 0.3777678 0.1494879 0.013038205 0.06766914 0.009339134
## 2      2 0.1980863 0.4130157 0.1495969 0.009385366 0.06155075 0.008390312
## 3      3 0.1889702 0.4656558 0.1435723 0.009611728 0.03700280 0.008884902
## 4      4 0.1882616 0.4697285 0.1436825 0.009994344 0.03384342 0.008142159
## 5      5 0.1841933 0.4921710 0.1417201 0.011991674 0.04868312 0.009469273
## 6      6 0.1836906 0.4947454 0.1412724 0.011580542 0.05086569 0.008098952
## 7      7 0.1819367 0.5038802 0.1405544 0.011323619 0.04721548 0.008132983
## 8      8 0.1815373 0.5057130 0.1394399 0.010978828 0.04708293 0.007602473
## 9      9 0.1807420 0.5093612 0.1390457 0.010185091 0.04485822 0.006754325
## 10     10 0.1807905 0.5091646 0.1393108 0.010052605 0.04357026 0.006941399
## 11     11 0.1808864 0.5086471 0.1394837 0.009905718 0.04480604 0.006538665
## 12     12 0.1812204 0.5069891 0.1397146 0.010475619 0.04806086 0.006904193
## 13     13 0.1811344 0.5073143 0.1397867 0.010223272 0.04627425 0.006816784
## 14     14 0.1808260 0.5089978 0.1395601 0.010537676 0.04727980 0.007043144
## 15     15 0.1807655 0.5094014 0.1395333 0.010362966 0.04689159 0.006909715
## 16     16 0.1808619 0.5089869 0.1396446 0.010010337 0.04627448 0.006591465
## 17     17 0.1808619 0.5089869 0.1396446 0.010010337 0.04627448 0.006591465
```

```
step.model$bestTune
```

```
##      nvmax
## 9         9
```

```
# creating a cool graph
nvmax.df <- step.model$results
graph_nvmax <- c(nvmax.df[["nvmax"]])
graph_rmse <- c(nvmax.df[["RMSE"]])
ggplot(data = data.frame(graph_nvmax, graph_rmse)) +
  geom_point(mapping = aes(x = graph_nvmax, y = graph_rmse),
    size = 3) +
  labs(title = "Backwards stepwise linear regression",
    x = "Maximum # of predictors",
    y = "Best RMSE") +
  theme_bw() +
  theme(axis.title=element_text(size=18), axis.text=element_text(size=18),
    plot.title = element_text(size = 18))
```

## Backwards stepwise linear regression



```
summary(step.model$finalModel) # tells us which variables to include
```

```
## Subset selection object
## 16 Variables (and intercept)
##           Forced in Forced out
## fg_pct      FALSE      FALSE
## fg3_pct      FALSE      FALSE
## ft_pct       FALSE      FALSE
## trb_per_g    FALSE      FALSE
## ast_per_g    FALSE      FALSE
## stl_per_g    FALSE      FALSE
## blk_per_g    FALSE      FALSE
## per         FALSE      FALSE
## ts_pct       FALSE      FALSE
## usg_pct      FALSE      FALSE
## ws          FALSE      FALSE
## bpm         FALSE      FALSE
## vorp        FALSE      FALSE
## age         FALSE      FALSE
## win_pct     FALSE      FALSE
## pts_per_g   FALSE      FALSE
## 1 subsets of each size up to 9
## Selection Algorithm: backward
##           fg_pct fg3_pct ft_pct trb_per_g ast_per_g stl_per_g blk_per_g per
## 1  ( 1 ) " " " " " " " " " " " " " "
```

```
## 2 ( 1 ) " " " " " " " " " " " "
## 3 ( 1 ) " " " " " " " " " " " "
## 4 ( 1 ) " " " " " " " " " " " "
## 5 ( 1 ) " " " " " " " " " " " "
## 6 ( 1 ) " " " " " " " " "*" " " "
## 7 ( 1 ) " " " " " " "*" "*" " " " "
## 8 ( 1 ) " " " " " "*" "*" "*" " " " "
## 9 ( 1 ) "*" " " " "*" "*" "*" " " " "
##      ts_pct usg_pct ws bpm vorp age win_pct pts_per_g
## 1 ( 1 ) " " " " " "*" " " " " " "
## 2 ( 1 ) " " " " " "*" " " " " "*" "
## 3 ( 1 ) " " " " " "*" " " " " "*" "*"
## 4 ( 1 ) " " " " "*" "*" " " " " "*" "*"
## 5 ( 1 ) "*" " " "*" "*" " " " " "*" "*"
## 6 ( 1 ) "*" " " "*" "*" " " " " "*" "*"
## 7 ( 1 ) "*" " " "*" "*" " " " " "*" "*"
## 8 ( 1 ) "*" " " "*" "*" " " " " "*" "*"
## 9 ( 1 ) "*" " " "*" "*" " " " " "*" "*"

```

```
coef(step.model$finalModel, id = 9)
```

```
## (Intercept)      fg_pct      trb_per_g      ast_per_g      stl_per_g
## -0.387387572  0.425193698  0.006473243  0.016045291 -0.054028684
##      ts_pct      ws      bpm      win_pct      pts_per_g
## -1.364878622  0.025383895  0.020137578  0.613492903  0.013981895

```

```
#building final linear model
```

```
bswr.mod <- lm(award_share ~ fg_pct + trb_per_g + ast_per_g + stl_per_g +
               ts_pct + ws + bpm + win_pct + pts_per_g,
               data = nba_csv.train)
summary(bswr.mod)
```

```
##
## Call:
## lm(formula = award_share ~ fg_pct + trb_per_g + ast_per_g + stl_per_g +
##      ts_pct + ws + bpm + win_pct + pts_per_g, data = nba_csv.train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -0.38158 -0.11672 -0.03796  0.09170  0.69670
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.387388   0.153987  -2.516  0.01213 *
## fg_pct       0.425194   0.260576   1.632  0.10323
## trb_per_g     0.006473   0.003485   1.858  0.06369 .
## ast_per_g     0.016045   0.003737   4.294 2.03e-05 ***
## stl_per_g    -0.054029   0.017149  -3.151  0.00171 **
## ts_pct      -1.364879   0.332031  -4.111 4.47e-05 ***
## ws           0.025384   0.003995   6.354 4.03e-10 ***
## bpm          0.020138   0.004676   4.307 1.92e-05 ***
## win_pct      0.613493   0.075165   8.162 1.81e-15 ***
## pts_per_g    0.013982   0.001792   7.803 2.53e-14 ***

```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.178 on 627 degrees of freedom
## Multiple R-squared:  0.5266, Adjusted R-squared:  0.5198
## F-statistic: 77.51 on 9 and 627 DF,  p-value: < 2.2e-16

vif(bswr.mod) # acceptable vifs

##      fg_pct trb_per_g ast_per_g stl_per_g      ts_pct      ws      bpm
## 2.768500 2.642912 2.258271 2.031006 3.389390 3.653558 3.194391
## win_pct pts_per_g
## 1.324580 1.709839

# testing for OSR2
pred.bswr <- predict(bswr.mod, newdata=nba_csv.test)
pred.bswr

##      1      2      3      4      5      6      7
## 0.5039477 0.5375057 0.1740437 0.3132550 0.1671717 0.2550683 0.1880855
##      8      9     10     11     12
## 0.2396639 0.1470319 0.2419638 0.1414766 0.1506761

nba_csv.test$award_share

## [1] 0.932 0.768 0.352 0.210 0.173 0.068 0.049 0.025 0.013 0.008 0.001
## [12] 0.001

SSE <- sum((nba_csv.test$award_share - pred.bswr)^2)
SST = sum((nba_csv.test$award_share - mean(nba_csv.train$award_share))^2)
OSR2 = 1 - SSE/SST
OSR2 # 0.5691312 # slight improvement!

## [1] 0.5691312
```

## Random forests – basic and cross-validated

```
##basic random forest model
set.seed(144)
mod.rf <- randomForest(award_share ~ ., data = nba_csv.train, mtry = 5, nodesize = 5, ntree = 500)
pred.rf <- predict(mod.rf, newdata = nba_csv.test) # just to illustrate

importance(mod.rf) #most important features: ws, vorp, win_pct, bpm

##      IncNodePurity
## fg_pct      1.0813085
## fg3_pct     0.8731366
## ft_pct      1.2082597
```

```
## trb_per_g      0.9318022
## ast_per_g      1.2053944
## stl_per_g      0.5994095
## blk_per_g      0.7338723
## per            5.0909337
## ts_pct         0.9842470
## usg_pct        1.8279452
## ws            9.0585303
## bpm           3.9519190
## vorp          5.5728829
## age           0.7868769
## win_pct        4.3633348
## pts_per_g      2.2574873
```

```
#cross validation on mtry
set.seed(849)
train.rf <- train(award_share ~ .,
                  data = nba_csv.train,
                  method = "rf",
                  tuneGrid = data.frame(mtry=1:16),
                  trControl = trainControl(method="cv",
                                           number=5, verboseIter = TRUE),
                  metric = "RMSE")
```

```
## + Fold1: mtry= 1
## - Fold1: mtry= 1
## + Fold1: mtry= 2
## - Fold1: mtry= 2
## + Fold1: mtry= 3
## - Fold1: mtry= 3
## + Fold1: mtry= 4
## - Fold1: mtry= 4
## + Fold1: mtry= 5
## - Fold1: mtry= 5
## + Fold1: mtry= 6
## - Fold1: mtry= 6
## + Fold1: mtry= 7
## - Fold1: mtry= 7
## + Fold1: mtry= 8
## - Fold1: mtry= 8
## + Fold1: mtry= 9
## - Fold1: mtry= 9
## + Fold1: mtry=10
## - Fold1: mtry=10
## + Fold1: mtry=11
## - Fold1: mtry=11
## + Fold1: mtry=12
## - Fold1: mtry=12
## + Fold1: mtry=13
## - Fold1: mtry=13
## + Fold1: mtry=14
## - Fold1: mtry=14
## + Fold1: mtry=15
## - Fold1: mtry=15
```

```
## + Fold1: mtry=16
## - Fold1: mtry=16
## + Fold2: mtry= 1
## - Fold2: mtry= 1
## + Fold2: mtry= 2
## - Fold2: mtry= 2
## + Fold2: mtry= 3
## - Fold2: mtry= 3
## + Fold2: mtry= 4
## - Fold2: mtry= 4
## + Fold2: mtry= 5
## - Fold2: mtry= 5
## + Fold2: mtry= 6
## - Fold2: mtry= 6
## + Fold2: mtry= 7
## - Fold2: mtry= 7
## + Fold2: mtry= 8
## - Fold2: mtry= 8
## + Fold2: mtry= 9
## - Fold2: mtry= 9
## + Fold2: mtry=10
## - Fold2: mtry=10
## + Fold2: mtry=11
## - Fold2: mtry=11
## + Fold2: mtry=12
## - Fold2: mtry=12
## + Fold2: mtry=13
## - Fold2: mtry=13
## + Fold2: mtry=14
## - Fold2: mtry=14
## + Fold2: mtry=15
## - Fold2: mtry=15
## + Fold2: mtry=16
## - Fold2: mtry=16
## + Fold3: mtry= 1
## - Fold3: mtry= 1
## + Fold3: mtry= 2
## - Fold3: mtry= 2
## + Fold3: mtry= 3
## - Fold3: mtry= 3
## + Fold3: mtry= 4
## - Fold3: mtry= 4
## + Fold3: mtry= 5
## - Fold3: mtry= 5
## + Fold3: mtry= 6
## - Fold3: mtry= 6
## + Fold3: mtry= 7
## - Fold3: mtry= 7
## + Fold3: mtry= 8
## - Fold3: mtry= 8
## + Fold3: mtry= 9
## - Fold3: mtry= 9
## + Fold3: mtry=10
## - Fold3: mtry=10
```



```
## + Fold3: mtry=11
## - Fold3: mtry=11
## + Fold3: mtry=12
## - Fold3: mtry=12
## + Fold3: mtry=13
## - Fold3: mtry=13
## + Fold3: mtry=14
## - Fold3: mtry=14
## + Fold3: mtry=15
## - Fold3: mtry=15
## + Fold3: mtry=16
## - Fold3: mtry=16
## + Fold4: mtry= 1
## - Fold4: mtry= 1
## + Fold4: mtry= 2
## - Fold4: mtry= 2
## + Fold4: mtry= 3
## - Fold4: mtry= 3
## + Fold4: mtry= 4
## - Fold4: mtry= 4
## + Fold4: mtry= 5
## - Fold4: mtry= 5
## + Fold4: mtry= 6
## - Fold4: mtry= 6
## + Fold4: mtry= 7
## - Fold4: mtry= 7
## + Fold4: mtry= 8
## - Fold4: mtry= 8
## + Fold4: mtry= 9
## - Fold4: mtry= 9
## + Fold4: mtry=10
## - Fold4: mtry=10
## + Fold4: mtry=11
## - Fold4: mtry=11
## + Fold4: mtry=12
## - Fold4: mtry=12
## + Fold4: mtry=13
## - Fold4: mtry=13
## + Fold4: mtry=14
## - Fold4: mtry=14
## + Fold4: mtry=15
## - Fold4: mtry=15
## + Fold4: mtry=16
## - Fold4: mtry=16
## + Fold5: mtry= 1
## - Fold5: mtry= 1
## + Fold5: mtry= 2
## - Fold5: mtry= 2
## + Fold5: mtry= 3
## - Fold5: mtry= 3
## + Fold5: mtry= 4
## - Fold5: mtry= 4
## + Fold5: mtry= 5
## - Fold5: mtry= 5
```

```

## + Fold5: mtry= 6
## - Fold5: mtry= 6
## + Fold5: mtry= 7
## - Fold5: mtry= 7
## + Fold5: mtry= 8
## - Fold5: mtry= 8
## + Fold5: mtry= 9
## - Fold5: mtry= 9
## + Fold5: mtry=10
## - Fold5: mtry=10
## + Fold5: mtry=11
## - Fold5: mtry=11
## + Fold5: mtry=12
## - Fold5: mtry=12
## + Fold5: mtry=13
## - Fold5: mtry=13
## + Fold5: mtry=14
## - Fold5: mtry=14
## + Fold5: mtry=15
## - Fold5: mtry=15
## + Fold5: mtry=16
## - Fold5: mtry=16
## Aggregating results
## Selecting tuning parameters
## Fitting mtry = 12 on full training set

```

```
train.rf #mtry=12
```

```

## Random Forest
##
## 637 samples
## 16 predictor
##
## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 509, 508, 510, 512, 509
## Resampling results across tuning parameters:
##
##   mtry  RMSE      Rsquared  MAE
##   1     0.1729006  0.5751842  0.1148861
##   2     0.1677458  0.5896253  0.1091635
##   3     0.1659232  0.5954343  0.1067127
##   4     0.1642990  0.6036294  0.1052672
##   5     0.1640126  0.6042265  0.1045772
##   6     0.1636150  0.6054612  0.1041081
##   7     0.1628583  0.6086627  0.1030604
##   8     0.1630133  0.6074687  0.1027573
##   9     0.1627223  0.6080682  0.1025852
##  10     0.1626249  0.6087426  0.1023939
##  11     0.1624621  0.6107147  0.1020349
##  12     0.1621851  0.6112369  0.1018550
##  13     0.1631822  0.6079965  0.1023682
##  14     0.1631774  0.6073143  0.1027458
##  15     0.1628195  0.6089939  0.1024110

```

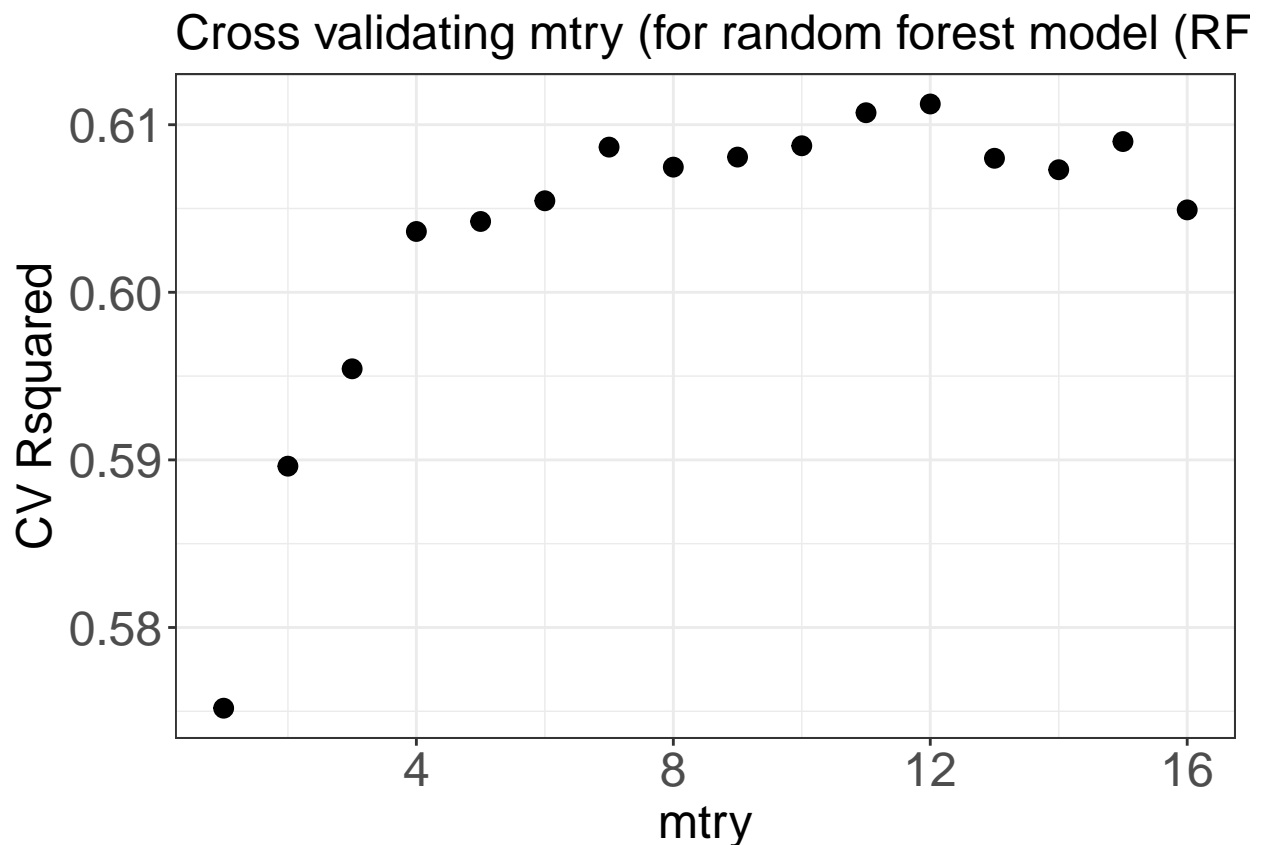
```
## 16 0.1637064 0.6049178 0.1031926
```

```
##
```

```
## RMSE was used to select the optimal model using the smallest value.
```

```
## The final value used for the model was mtry = 12.
```

```
best.rf <- train.rf$finalModel  
pred.best.rf <- predict(best.rf, newdata = nba_csv.test) # can use same model matrix  
  
ggplot(train.rf$results, aes(x = mtry, y = Rsquared)) +  
  geom_point(size = 3) +  
  labs(y = "CV Rsquared", x = "mtry",  
       title = "Cross validating mtry (for random forest model (RF))" +  
  theme_bw() +  
  theme(axis.title=element_text(size=18), axis.text=element_text(size=18),  
        plot.title = element_text(size = 18))
```



```
SSE = sum((nba_csv.test$award_share - pred.best.rf)^2)  
SST = sum((nba_csv.test$award_share - mean(nba_csv.train$award_share))^2)  
OSR2 = 1 - SSE/SST  
OSR2 #0.7677164
```

```
## [1] 0.7677164
```

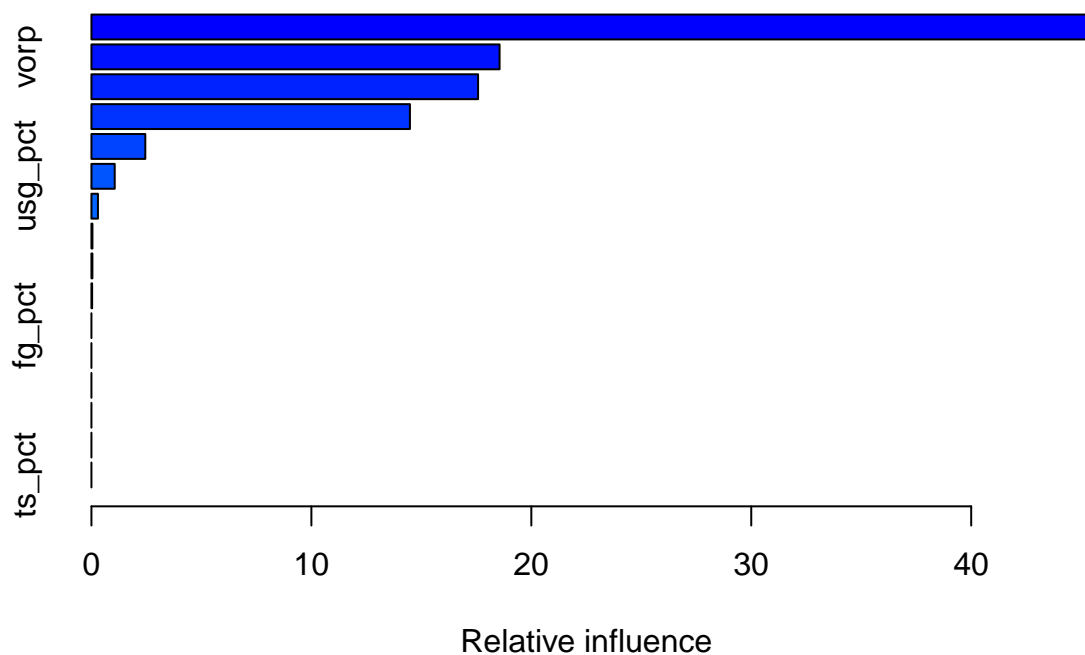
## Boosting – basic and cross-validated

```
#boosting model
mod.boost <- gbm(award_share ~ .,
                 data = nba_csv.train,
                 distribution = "gaussian",
                 n.trees = 1000,
                 shrinkage = 0.001,
                 interaction.depth = 2)

pred.boost <- predict(mod.boost, newdata = nba_csv.test, n.trees=1000)

pred.boost.earlier <- predict(mod.boost, newdata = nba_csv.test, n.trees=330)

summary(mod.boost) #most influential features are ws, vorp, per, win_pct
```



```
##          var      rel.inf
## ws          ws 45.46286772
## vorp        vorp 18.55601031
## per          per 17.57885978
## win_pct     win_pct 14.47977396
## bpm          bpm  2.44925995
## usg_pct     usg_pct  1.05795822
## pts_per_g   pts_per_g  0.29577981
```

```
## age          age  0.04456713
## ft_pct      ft_pct 0.04326546
## ast_per_g   ast_per_g 0.03165767
## fg_pct      fg_pct 0.00000000
## fg3_pct     fg3_pct 0.00000000
## trb_per_g   trb_per_g 0.00000000
## stl_per_g   stl_per_g 0.00000000
## blk_per_g   blk_per_g 0.00000000
## ts_pct      ts_pct 0.00000000
```

*#cross validation on n.trees and interaction depth*

```
tGrid = expand.grid(n.trees = (1:75)*500, interaction.depth = c(1,2,4,6,8,10),
                    shrinkage = 0.001, n.minobsinnode = 10)
```

```
set.seed(849)
```

```
train.boost <- train(award_share ~ .,
                     data = nba_csv.train,
                     method = "gbm",
                     tuneGrid = tGrid,
                     trControl = trainControl(method="cv", number=5,
                                                verboseIter = TRUE),
                     metric = "RMSE",
                     distribution = "gaussian")
```

```
## + Fold1: shrinkage=0.001, interaction.depth= 1, n.minobsinnode=10, n.trees=37500
```

## Iter	TrainDeviance	ValidDeviance	StepSize	Improve
## 1	0.0665	nan	0.0010	0.0001
## 2	0.0665	nan	0.0010	0.0000
## 3	0.0664	nan	0.0010	0.0000
## 4	0.0663	nan	0.0010	0.0000
## 5	0.0663	nan	0.0010	0.0000
## 6	0.0662	nan	0.0010	0.0000
## 7	0.0662	nan	0.0010	0.0001
## 8	0.0662	nan	0.0010	0.0000
## 9	0.0661	nan	0.0010	0.0000
## 10	0.0661	nan	0.0010	0.0000
## 20	0.0656	nan	0.0010	0.0000
## 40	0.0646	nan	0.0010	0.0000
## 60	0.0637	nan	0.0010	0.0000
## 80	0.0628	nan	0.0010	0.0000
## 100	0.0620	nan	0.0010	0.0000
## 120	0.0611	nan	0.0010	0.0000
## 140	0.0603	nan	0.0010	0.0001
## 160	0.0595	nan	0.0010	0.0000
## 180	0.0587	nan	0.0010	0.0000
## 200	0.0579	nan	0.0010	0.0000
## 220	0.0572	nan	0.0010	0.0000
## 240	0.0565	nan	0.0010	0.0000
## 260	0.0558	nan	0.0010	0.0000
## 280	0.0551	nan	0.0010	0.0000
## 300	0.0545	nan	0.0010	0.0000
## 320	0.0539	nan	0.0010	0.0000
## 340	0.0533	nan	0.0010	0.0000
## 360	0.0527	nan	0.0010	0.0000

##	380	0.0521	nan	0.0010	0.0000
##	400	0.0516	nan	0.0010	0.0000
##	420	0.0510	nan	0.0010	0.0000
##	440	0.0505	nan	0.0010	0.0000
##	460	0.0500	nan	0.0010	0.0000
##	480	0.0495	nan	0.0010	0.0000
##	500	0.0490	nan	0.0010	0.0000
##	520	0.0486	nan	0.0010	0.0000
##	540	0.0481	nan	0.0010	0.0000
##	560	0.0477	nan	0.0010	0.0000
##	580	0.0472	nan	0.0010	0.0000
##	600	0.0468	nan	0.0010	0.0000
##	620	0.0464	nan	0.0010	0.0000
##	640	0.0460	nan	0.0010	0.0000
##	660	0.0456	nan	0.0010	0.0000
##	680	0.0452	nan	0.0010	0.0000
##	700	0.0449	nan	0.0010	0.0000
##	720	0.0445	nan	0.0010	0.0000
##	740	0.0442	nan	0.0010	0.0000
##	760	0.0438	nan	0.0010	0.0000
##	780	0.0435	nan	0.0010	0.0000
##	800	0.0432	nan	0.0010	0.0000
##	820	0.0428	nan	0.0010	0.0000
##	840	0.0425	nan	0.0010	0.0000
##	860	0.0422	nan	0.0010	0.0000
##	880	0.0419	nan	0.0010	0.0000
##	900	0.0417	nan	0.0010	0.0000
##	920	0.0414	nan	0.0010	0.0000
##	940	0.0411	nan	0.0010	0.0000
##	960	0.0408	nan	0.0010	0.0000
##	980	0.0406	nan	0.0010	0.0000
##	1000	0.0403	nan	0.0010	0.0000
##	1020	0.0400	nan	0.0010	0.0000
##	1040	0.0398	nan	0.0010	0.0000
##	1060	0.0396	nan	0.0010	0.0000
##	1080	0.0393	nan	0.0010	0.0000
##	1100	0.0391	nan	0.0010	0.0000
##	1120	0.0388	nan	0.0010	0.0000
##	1140	0.0386	nan	0.0010	0.0000
##	1160	0.0384	nan	0.0010	0.0000
##	1180	0.0382	nan	0.0010	0.0000
##	1200	0.0380	nan	0.0010	0.0000
##	1220	0.0378	nan	0.0010	0.0000
##	1240	0.0376	nan	0.0010	0.0000
##	1260	0.0374	nan	0.0010	0.0000
##	1280	0.0372	nan	0.0010	0.0000
##	1300	0.0370	nan	0.0010	0.0000
##	1320	0.0368	nan	0.0010	0.0000
##	1340	0.0366	nan	0.0010	0.0000
##	1360	0.0365	nan	0.0010	0.0000
##	1380	0.0363	nan	0.0010	0.0000
##	1400	0.0361	nan	0.0010	0.0000
##	1420	0.0359	nan	0.0010	0.0000
##	1440	0.0358	nan	0.0010	0.0000

##	1460	0.0356	nan	0.0010	0.0000
##	1480	0.0355	nan	0.0010	0.0000
##	1500	0.0353	nan	0.0010	0.0000
##	1520	0.0351	nan	0.0010	0.0000
##	1540	0.0350	nan	0.0010	0.0000
##	1560	0.0348	nan	0.0010	0.0000
##	1580	0.0347	nan	0.0010	0.0000
##	1600	0.0345	nan	0.0010	0.0000
##	1620	0.0344	nan	0.0010	0.0000
##	1640	0.0342	nan	0.0010	0.0000
##	1660	0.0341	nan	0.0010	0.0000
##	1680	0.0339	nan	0.0010	0.0000
##	1700	0.0338	nan	0.0010	0.0000
##	1720	0.0337	nan	0.0010	0.0000
##	1740	0.0335	nan	0.0010	0.0000
##	1760	0.0334	nan	0.0010	0.0000
##	1780	0.0333	nan	0.0010	0.0000
##	1800	0.0331	nan	0.0010	0.0000
##	1820	0.0330	nan	0.0010	0.0000
##	1840	0.0329	nan	0.0010	0.0000
##	1860	0.0328	nan	0.0010	0.0000
##	1880	0.0327	nan	0.0010	0.0000
##	1900	0.0325	nan	0.0010	0.0000
##	1920	0.0324	nan	0.0010	0.0000
##	1940	0.0323	nan	0.0010	0.0000
##	1960	0.0322	nan	0.0010	0.0000
##	1980	0.0321	nan	0.0010	0.0000
##	2000	0.0320	nan	0.0010	0.0000
##	2020	0.0319	nan	0.0010	-0.0000
##	2040	0.0318	nan	0.0010	0.0000
##	2060	0.0317	nan	0.0010	0.0000
##	2080	0.0316	nan	0.0010	0.0000
##	2100	0.0315	nan	0.0010	0.0000
##	2120	0.0314	nan	0.0010	0.0000
##	2140	0.0313	nan	0.0010	0.0000
##	2160	0.0312	nan	0.0010	0.0000
##	2180	0.0311	nan	0.0010	0.0000
##	2200	0.0310	nan	0.0010	0.0000
##	2220	0.0309	nan	0.0010	0.0000
##	2240	0.0308	nan	0.0010	0.0000
##	2260	0.0307	nan	0.0010	0.0000
##	2280	0.0306	nan	0.0010	0.0000
##	2300	0.0305	nan	0.0010	0.0000
##	2320	0.0304	nan	0.0010	0.0000
##	2340	0.0304	nan	0.0010	0.0000
##	2360	0.0303	nan	0.0010	0.0000
##	2380	0.0302	nan	0.0010	0.0000
##	2400	0.0301	nan	0.0010	0.0000
##	2420	0.0300	nan	0.0010	0.0000
##	2440	0.0300	nan	0.0010	0.0000
##	2460	0.0299	nan	0.0010	0.0000
##	2480	0.0298	nan	0.0010	0.0000
##	2500	0.0297	nan	0.0010	0.0000
##	2520	0.0297	nan	0.0010	0.0000

##	2540	0.0296	nan	0.0010	0.0000
##	2560	0.0295	nan	0.0010	0.0000
##	2580	0.0294	nan	0.0010	0.0000
##	2600	0.0294	nan	0.0010	0.0000
##	2620	0.0293	nan	0.0010	0.0000
##	2640	0.0292	nan	0.0010	0.0000
##	2660	0.0291	nan	0.0010	0.0000
##	2680	0.0291	nan	0.0010	0.0000
##	2700	0.0290	nan	0.0010	0.0000
##	2720	0.0289	nan	0.0010	0.0000
##	2740	0.0289	nan	0.0010	0.0000
##	2760	0.0288	nan	0.0010	0.0000
##	2780	0.0288	nan	0.0010	0.0000
##	2800	0.0287	nan	0.0010	0.0000
##	2820	0.0286	nan	0.0010	0.0000
##	2840	0.0286	nan	0.0010	0.0000
##	2860	0.0285	nan	0.0010	0.0000
##	2880	0.0285	nan	0.0010	0.0000
##	2900	0.0284	nan	0.0010	0.0000
##	2920	0.0284	nan	0.0010	0.0000
##	2940	0.0283	nan	0.0010	0.0000
##	2960	0.0282	nan	0.0010	0.0000
##	2980	0.0282	nan	0.0010	0.0000
##	3000	0.0281	nan	0.0010	0.0000
##	3020	0.0281	nan	0.0010	0.0000
##	3040	0.0280	nan	0.0010	0.0000
##	3060	0.0280	nan	0.0010	0.0000
##	3080	0.0279	nan	0.0010	0.0000
##	3100	0.0279	nan	0.0010	0.0000
##	3120	0.0278	nan	0.0010	0.0000
##	3140	0.0278	nan	0.0010	0.0000
##	3160	0.0277	nan	0.0010	0.0000
##	3180	0.0277	nan	0.0010	0.0000
##	3200	0.0276	nan	0.0010	0.0000
##	3220	0.0276	nan	0.0010	-0.0000
##	3240	0.0275	nan	0.0010	0.0000
##	3260	0.0275	nan	0.0010	0.0000
##	3280	0.0274	nan	0.0010	-0.0000
##	3300	0.0274	nan	0.0010	0.0000
##	3320	0.0274	nan	0.0010	0.0000
##	3340	0.0273	nan	0.0010	0.0000
##	3360	0.0273	nan	0.0010	0.0000
##	3380	0.0272	nan	0.0010	0.0000
##	3400	0.0272	nan	0.0010	0.0000
##	3420	0.0272	nan	0.0010	0.0000
##	3440	0.0271	nan	0.0010	-0.0000
##	3460	0.0271	nan	0.0010	0.0000
##	3480	0.0270	nan	0.0010	0.0000
##	3500	0.0270	nan	0.0010	-0.0000
##	3520	0.0270	nan	0.0010	-0.0000
##	3540	0.0269	nan	0.0010	0.0000
##	3560	0.0269	nan	0.0010	0.0000
##	3580	0.0269	nan	0.0010	0.0000
##	3600	0.0268	nan	0.0010	0.0000



##	3620	0.0268	nan	0.0010	0.0000
##	3640	0.0267	nan	0.0010	0.0000
##	3660	0.0267	nan	0.0010	0.0000
##	3680	0.0267	nan	0.0010	-0.0000
##	3700	0.0266	nan	0.0010	0.0000
##	3720	0.0266	nan	0.0010	0.0000
##	3740	0.0266	nan	0.0010	0.0000
##	3760	0.0265	nan	0.0010	0.0000
##	3780	0.0265	nan	0.0010	0.0000
##	3800	0.0265	nan	0.0010	0.0000
##	3820	0.0265	nan	0.0010	0.0000
##	3840	0.0264	nan	0.0010	0.0000
##	3860	0.0264	nan	0.0010	0.0000
##	3880	0.0264	nan	0.0010	-0.0000
##	3900	0.0263	nan	0.0010	-0.0000
##	3920	0.0263	nan	0.0010	0.0000
##	3940	0.0263	nan	0.0010	0.0000
##	3960	0.0262	nan	0.0010	-0.0000
##	3980	0.0262	nan	0.0010	0.0000
##	4000	0.0262	nan	0.0010	0.0000
##	4020	0.0262	nan	0.0010	-0.0000
##	4040	0.0261	nan	0.0010	0.0000
##	4060	0.0261	nan	0.0010	0.0000
##	4080	0.0261	nan	0.0010	-0.0000
##	4100	0.0260	nan	0.0010	-0.0000
##	4120	0.0260	nan	0.0010	-0.0000
##	4140	0.0260	nan	0.0010	-0.0000
##	4160	0.0260	nan	0.0010	-0.0000
##	4180	0.0259	nan	0.0010	0.0000
##	4200	0.0259	nan	0.0010	0.0000
##	4220	0.0259	nan	0.0010	0.0000
##	4240	0.0259	nan	0.0010	-0.0000
##	4260	0.0258	nan	0.0010	-0.0000
##	4280	0.0258	nan	0.0010	-0.0000
##	4300	0.0258	nan	0.0010	0.0000
##	4320	0.0258	nan	0.0010	-0.0000
##	4340	0.0258	nan	0.0010	-0.0000
##	4360	0.0257	nan	0.0010	0.0000
##	4380	0.0257	nan	0.0010	-0.0000
##	4400	0.0257	nan	0.0010	-0.0000
##	4420	0.0257	nan	0.0010	0.0000
##	4440	0.0256	nan	0.0010	0.0000
##	4460	0.0256	nan	0.0010	-0.0000
##	4480	0.0256	nan	0.0010	-0.0000
##	4500	0.0256	nan	0.0010	0.0000
##	4520	0.0256	nan	0.0010	0.0000
##	4540	0.0255	nan	0.0010	0.0000
##	4560	0.0255	nan	0.0010	-0.0000
##	4580	0.0255	nan	0.0010	-0.0000
##	4600	0.0255	nan	0.0010	-0.0000
##	4620	0.0254	nan	0.0010	0.0000
##	4640	0.0254	nan	0.0010	-0.0000
##	4660	0.0254	nan	0.0010	0.0000
##	4680	0.0254	nan	0.0010	-0.0000

##	4700	0.0254	nan	0.0010	-0.0000
##	4720	0.0253	nan	0.0010	0.0000
##	4740	0.0253	nan	0.0010	0.0000
##	4760	0.0253	nan	0.0010	-0.0000
##	4780	0.0253	nan	0.0010	0.0000
##	4800	0.0253	nan	0.0010	-0.0000
##	4820	0.0253	nan	0.0010	-0.0000
##	4840	0.0252	nan	0.0010	0.0000
##	4860	0.0252	nan	0.0010	0.0000
##	4880	0.0252	nan	0.0010	0.0000
##	4900	0.0252	nan	0.0010	0.0000
##	4920	0.0252	nan	0.0010	-0.0000
##	4940	0.0251	nan	0.0010	-0.0000
##	4960	0.0251	nan	0.0010	-0.0000
##	4980	0.0251	nan	0.0010	0.0000
##	5000	0.0251	nan	0.0010	-0.0000
##	5020	0.0251	nan	0.0010	-0.0000
##	5040	0.0251	nan	0.0010	-0.0000
##	5060	0.0250	nan	0.0010	-0.0000
##	5080	0.0250	nan	0.0010	-0.0000
##	5100	0.0250	nan	0.0010	-0.0000
##	5120	0.0250	nan	0.0010	-0.0000
##	5140	0.0250	nan	0.0010	-0.0000
##	5160	0.0250	nan	0.0010	-0.0000
##	5180	0.0250	nan	0.0010	0.0000
##	5200	0.0249	nan	0.0010	-0.0000
##	5220	0.0249	nan	0.0010	-0.0000
##	5240	0.0249	nan	0.0010	-0.0000
##	5260	0.0249	nan	0.0010	-0.0000
##	5280	0.0249	nan	0.0010	0.0000
##	5300	0.0249	nan	0.0010	-0.0000
##	5320	0.0248	nan	0.0010	-0.0000
##	5340	0.0248	nan	0.0010	-0.0000
##	5360	0.0248	nan	0.0010	0.0000
##	5380	0.0248	nan	0.0010	-0.0000
##	5400	0.0248	nan	0.0010	-0.0000
##	5420	0.0248	nan	0.0010	0.0000
##	5440	0.0248	nan	0.0010	-0.0000
##	5460	0.0247	nan	0.0010	-0.0000
##	5480	0.0247	nan	0.0010	-0.0000
##	5500	0.0247	nan	0.0010	-0.0000
##	5520	0.0247	nan	0.0010	-0.0000
##	5540	0.0247	nan	0.0010	-0.0000
##	5560	0.0247	nan	0.0010	-0.0000
##	5580	0.0247	nan	0.0010	-0.0000
##	5600	0.0246	nan	0.0010	-0.0000
##	5620	0.0246	nan	0.0010	-0.0000
##	5640	0.0246	nan	0.0010	-0.0000
##	5660	0.0246	nan	0.0010	-0.0000
##	5680	0.0246	nan	0.0010	-0.0000
##	5700	0.0246	nan	0.0010	-0.0000
##	5720	0.0246	nan	0.0010	0.0000
##	5740	0.0245	nan	0.0010	-0.0000
##	5760	0.0245	nan	0.0010	0.0000

```

## 36600      0.0003      nan    0.0010   -0.0000
## 36620      0.0003      nan    0.0010   -0.0000
## 36640      0.0003      nan    0.0010   -0.0000
## 36660      0.0003      nan    0.0010   -0.0000
## 36680      0.0003      nan    0.0010   -0.0000
## 36700      0.0003      nan    0.0010   -0.0000
## 36720      0.0003      nan    0.0010   -0.0000
## 36740      0.0003      nan    0.0010   -0.0000
## 36760      0.0003      nan    0.0010   -0.0000
## 36780      0.0003      nan    0.0010   -0.0000
## 36800      0.0003      nan    0.0010   -0.0000
## 36820      0.0003      nan    0.0010   -0.0000
## 36840      0.0003      nan    0.0010   -0.0000
## 36860      0.0003      nan    0.0010   -0.0000
## 36880      0.0003      nan    0.0010   -0.0000
## 36900      0.0003      nan    0.0010   -0.0000
## 36920      0.0003      nan    0.0010   -0.0000
## 36940      0.0003      nan    0.0010   -0.0000
## 36960      0.0003      nan    0.0010   -0.0000
## 36980      0.0003      nan    0.0010   -0.0000
## 37000      0.0003      nan    0.0010   -0.0000
## 37020      0.0003      nan    0.0010   -0.0000
## 37040      0.0003      nan    0.0010   -0.0000
## 37060      0.0003      nan    0.0010   -0.0000
## 37080      0.0003      nan    0.0010   -0.0000
## 37100      0.0003      nan    0.0010   -0.0000
## 37120      0.0003      nan    0.0010   -0.0000
## 37140      0.0003      nan    0.0010   -0.0000
## 37160      0.0003      nan    0.0010   -0.0000
## 37180      0.0003      nan    0.0010   -0.0000
## 37200      0.0003      nan    0.0010   -0.0000
## 37220      0.0003      nan    0.0010   -0.0000
## 37240      0.0003      nan    0.0010   -0.0000
## 37260      0.0003      nan    0.0010   -0.0000
## 37280      0.0003      nan    0.0010   -0.0000
## 37300      0.0003      nan    0.0010   -0.0000
## 37320      0.0003      nan    0.0010   -0.0000
## 37340      0.0003      nan    0.0010   -0.0000
## 37360      0.0003      nan    0.0010   -0.0000
## 37380      0.0003      nan    0.0010   -0.0000
## 37400      0.0003      nan    0.0010   -0.0000
## 37420      0.0003      nan    0.0010   -0.0000
## 37440      0.0003      nan    0.0010   -0.0000
## 37460      0.0003      nan    0.0010   -0.0000
## 37480      0.0003      nan    0.0010   -0.0000
## 37500      0.0003      nan    0.0010   -0.0000
##
## - Fold5: shrinkage=0.001, interaction.depth=10, n.minobsinnode=10, n.trees=37500
## Aggregating results
## Selecting tuning parameters
## Fitting n.trees = 4000, interaction.depth = 4, shrinkage = 0.001, n.minobsinnode = 10 on full training set
## Iter  TrainDeviance  ValidDeviance  StepSize  Improve
##      1          0.0658          nan    0.0010    0.0001
##      2          0.0657          nan    0.0010    0.0001

```

##	3	0.0657	nan	0.0010	0.0001
##	4	0.0656	nan	0.0010	0.0001
##	5	0.0655	nan	0.0010	0.0001
##	6	0.0655	nan	0.0010	0.0001
##	7	0.0654	nan	0.0010	0.0001
##	8	0.0653	nan	0.0010	0.0001
##	9	0.0652	nan	0.0010	0.0001
##	10	0.0652	nan	0.0010	0.0001
##	20	0.0644	nan	0.0010	0.0001
##	40	0.0631	nan	0.0010	0.0000
##	60	0.0617	nan	0.0010	0.0001
##	80	0.0604	nan	0.0010	0.0001
##	100	0.0592	nan	0.0010	0.0000
##	120	0.0580	nan	0.0010	0.0000
##	140	0.0568	nan	0.0010	0.0001
##	160	0.0556	nan	0.0010	0.0001
##	180	0.0546	nan	0.0010	0.0000
##	200	0.0535	nan	0.0010	0.0000
##	220	0.0524	nan	0.0010	0.0000
##	240	0.0514	nan	0.0010	0.0000
##	260	0.0505	nan	0.0010	0.0000
##	280	0.0495	nan	0.0010	0.0000
##	300	0.0486	nan	0.0010	0.0000
##	320	0.0477	nan	0.0010	0.0000
##	340	0.0469	nan	0.0010	0.0000
##	360	0.0461	nan	0.0010	0.0000
##	380	0.0453	nan	0.0010	0.0000
##	400	0.0445	nan	0.0010	0.0000
##	420	0.0437	nan	0.0010	0.0000
##	440	0.0430	nan	0.0010	0.0000
##	460	0.0423	nan	0.0010	0.0000
##	480	0.0416	nan	0.0010	0.0000
##	500	0.0409	nan	0.0010	0.0000
##	520	0.0403	nan	0.0010	0.0000
##	540	0.0397	nan	0.0010	0.0000
##	560	0.0390	nan	0.0010	0.0000
##	580	0.0385	nan	0.0010	0.0000
##	600	0.0379	nan	0.0010	0.0000
##	620	0.0374	nan	0.0010	0.0000
##	640	0.0368	nan	0.0010	0.0000
##	660	0.0363	nan	0.0010	0.0000
##	680	0.0358	nan	0.0010	0.0000
##	700	0.0353	nan	0.0010	0.0000
##	720	0.0348	nan	0.0010	0.0000
##	740	0.0344	nan	0.0010	0.0000
##	760	0.0339	nan	0.0010	0.0000
##	780	0.0335	nan	0.0010	0.0000
##	800	0.0331	nan	0.0010	0.0000
##	820	0.0326	nan	0.0010	0.0000
##	840	0.0322	nan	0.0010	0.0000
##	860	0.0318	nan	0.0010	0.0000
##	880	0.0315	nan	0.0010	0.0000
##	900	0.0311	nan	0.0010	0.0000
##	920	0.0308	nan	0.0010	0.0000

##	940	0.0304	nan	0.0010	0.0000
##	960	0.0301	nan	0.0010	0.0000
##	980	0.0297	nan	0.0010	0.0000
##	1000	0.0294	nan	0.0010	0.0000
##	1020	0.0291	nan	0.0010	0.0000
##	1040	0.0288	nan	0.0010	0.0000
##	1060	0.0285	nan	0.0010	0.0000
##	1080	0.0282	nan	0.0010	0.0000
##	1100	0.0280	nan	0.0010	0.0000
##	1120	0.0277	nan	0.0010	0.0000
##	1140	0.0274	nan	0.0010	0.0000
##	1160	0.0272	nan	0.0010	0.0000
##	1180	0.0269	nan	0.0010	0.0000
##	1200	0.0267	nan	0.0010	0.0000
##	1220	0.0264	nan	0.0010	0.0000
##	1240	0.0262	nan	0.0010	0.0000
##	1260	0.0260	nan	0.0010	0.0000
##	1280	0.0257	nan	0.0010	0.0000
##	1300	0.0255	nan	0.0010	0.0000
##	1320	0.0253	nan	0.0010	0.0000
##	1340	0.0251	nan	0.0010	0.0000
##	1360	0.0249	nan	0.0010	0.0000
##	1380	0.0247	nan	0.0010	0.0000
##	1400	0.0245	nan	0.0010	0.0000
##	1420	0.0243	nan	0.0010	0.0000
##	1440	0.0242	nan	0.0010	0.0000
##	1460	0.0240	nan	0.0010	0.0000
##	1480	0.0238	nan	0.0010	0.0000
##	1500	0.0236	nan	0.0010	0.0000
##	1520	0.0235	nan	0.0010	0.0000
##	1540	0.0233	nan	0.0010	0.0000
##	1560	0.0231	nan	0.0010	0.0000
##	1580	0.0230	nan	0.0010	0.0000
##	1600	0.0228	nan	0.0010	0.0000
##	1620	0.0227	nan	0.0010	0.0000
##	1640	0.0225	nan	0.0010	0.0000
##	1660	0.0224	nan	0.0010	0.0000
##	1680	0.0222	nan	0.0010	0.0000
##	1700	0.0221	nan	0.0010	0.0000
##	1720	0.0220	nan	0.0010	0.0000
##	1740	0.0218	nan	0.0010	0.0000
##	1760	0.0217	nan	0.0010	0.0000
##	1780	0.0216	nan	0.0010	0.0000
##	1800	0.0215	nan	0.0010	0.0000
##	1820	0.0214	nan	0.0010	0.0000
##	1840	0.0212	nan	0.0010	0.0000
##	1860	0.0211	nan	0.0010	0.0000
##	1880	0.0210	nan	0.0010	0.0000
##	1900	0.0209	nan	0.0010	0.0000
##	1920	0.0208	nan	0.0010	0.0000
##	1940	0.0207	nan	0.0010	0.0000
##	1960	0.0206	nan	0.0010	0.0000
##	1980	0.0205	nan	0.0010	0.0000
##	2000	0.0204	nan	0.0010	0.0000

##	2020	0.0203	nan	0.0010	-0.0000
##	2040	0.0202	nan	0.0010	0.0000
##	2060	0.0201	nan	0.0010	0.0000
##	2080	0.0200	nan	0.0010	0.0000
##	2100	0.0199	nan	0.0010	0.0000
##	2120	0.0198	nan	0.0010	0.0000
##	2140	0.0197	nan	0.0010	0.0000
##	2160	0.0196	nan	0.0010	0.0000
##	2180	0.0195	nan	0.0010	0.0000
##	2200	0.0194	nan	0.0010	0.0000
##	2220	0.0194	nan	0.0010	0.0000
##	2240	0.0193	nan	0.0010	0.0000
##	2260	0.0192	nan	0.0010	-0.0000
##	2280	0.0191	nan	0.0010	0.0000
##	2300	0.0190	nan	0.0010	-0.0000
##	2320	0.0190	nan	0.0010	0.0000
##	2340	0.0189	nan	0.0010	0.0000
##	2360	0.0188	nan	0.0010	0.0000
##	2380	0.0188	nan	0.0010	0.0000
##	2400	0.0187	nan	0.0010	0.0000
##	2420	0.0186	nan	0.0010	0.0000
##	2440	0.0185	nan	0.0010	0.0000
##	2460	0.0185	nan	0.0010	0.0000
##	2480	0.0184	nan	0.0010	-0.0000
##	2500	0.0183	nan	0.0010	-0.0000
##	2520	0.0183	nan	0.0010	0.0000
##	2540	0.0182	nan	0.0010	0.0000
##	2560	0.0182	nan	0.0010	0.0000
##	2580	0.0181	nan	0.0010	-0.0000
##	2600	0.0180	nan	0.0010	0.0000
##	2620	0.0180	nan	0.0010	-0.0000
##	2640	0.0179	nan	0.0010	0.0000
##	2660	0.0179	nan	0.0010	-0.0000
##	2680	0.0178	nan	0.0010	-0.0000
##	2700	0.0178	nan	0.0010	-0.0000
##	2720	0.0177	nan	0.0010	-0.0000
##	2740	0.0176	nan	0.0010	-0.0000
##	2760	0.0176	nan	0.0010	-0.0000
##	2780	0.0175	nan	0.0010	0.0000
##	2800	0.0175	nan	0.0010	0.0000
##	2820	0.0174	nan	0.0010	0.0000
##	2840	0.0174	nan	0.0010	-0.0000
##	2860	0.0173	nan	0.0010	-0.0000
##	2880	0.0173	nan	0.0010	0.0000
##	2900	0.0172	nan	0.0010	-0.0000
##	2920	0.0172	nan	0.0010	0.0000
##	2940	0.0171	nan	0.0010	-0.0000
##	2960	0.0171	nan	0.0010	0.0000
##	2980	0.0170	nan	0.0010	-0.0000
##	3000	0.0170	nan	0.0010	-0.0000
##	3020	0.0169	nan	0.0010	-0.0000
##	3040	0.0169	nan	0.0010	-0.0000
##	3060	0.0168	nan	0.0010	0.0000
##	3080	0.0168	nan	0.0010	-0.0000

##	3100	0.0168	nan	0.0010	-0.0000
##	3120	0.0167	nan	0.0010	0.0000
##	3140	0.0167	nan	0.0010	-0.0000
##	3160	0.0166	nan	0.0010	-0.0000
##	3180	0.0166	nan	0.0010	-0.0000
##	3200	0.0165	nan	0.0010	-0.0000
##	3220	0.0165	nan	0.0010	-0.0000
##	3240	0.0165	nan	0.0010	0.0000
##	3260	0.0164	nan	0.0010	-0.0000
##	3280	0.0164	nan	0.0010	-0.0000
##	3300	0.0163	nan	0.0010	-0.0000
##	3320	0.0163	nan	0.0010	-0.0000
##	3340	0.0163	nan	0.0010	-0.0000
##	3360	0.0162	nan	0.0010	-0.0000
##	3380	0.0162	nan	0.0010	0.0000
##	3400	0.0161	nan	0.0010	-0.0000
##	3420	0.0161	nan	0.0010	-0.0000
##	3440	0.0161	nan	0.0010	-0.0000
##	3460	0.0160	nan	0.0010	0.0000
##	3480	0.0160	nan	0.0010	-0.0000
##	3500	0.0160	nan	0.0010	-0.0000
##	3520	0.0159	nan	0.0010	-0.0000
##	3540	0.0159	nan	0.0010	-0.0000
##	3560	0.0158	nan	0.0010	-0.0000
##	3580	0.0158	nan	0.0010	-0.0000
##	3600	0.0158	nan	0.0010	-0.0000
##	3620	0.0157	nan	0.0010	-0.0000
##	3640	0.0157	nan	0.0010	0.0000
##	3660	0.0157	nan	0.0010	0.0000
##	3680	0.0156	nan	0.0010	-0.0000
##	3700	0.0156	nan	0.0010	-0.0000
##	3720	0.0156	nan	0.0010	-0.0000
##	3740	0.0155	nan	0.0010	-0.0000
##	3760	0.0155	nan	0.0010	0.0000
##	3780	0.0155	nan	0.0010	-0.0000
##	3800	0.0154	nan	0.0010	-0.0000
##	3820	0.0154	nan	0.0010	-0.0000
##	3840	0.0154	nan	0.0010	-0.0000
##	3860	0.0153	nan	0.0010	-0.0000
##	3880	0.0153	nan	0.0010	-0.0000
##	3900	0.0153	nan	0.0010	-0.0000
##	3920	0.0152	nan	0.0010	-0.0000
##	3940	0.0152	nan	0.0010	-0.0000
##	3960	0.0152	nan	0.0010	-0.0000
##	3980	0.0152	nan	0.0010	0.0000
##	4000	0.0151	nan	0.0010	-0.0000

```
train.boost #ntrees=4000,interaction.depth=4
```

```
## Stochastic Gradient Boosting
##
## 637 samples
## 16 predictor
##
```

```

## No pre-processing
## Resampling: Cross-Validated (5 fold)
## Summary of sample sizes: 509, 508, 510, 512, 509
## Resampling results across tuning parameters:
##
##   interaction.depth  n.trees  RMSE      Rsquared  MAE
##   1                  500      0.2214558  0.4792111  0.16511538
##   1                  1000     0.2020549  0.4969852  0.14811466
##   1                  1500     0.1907759  0.5191214  0.13683499
##   1                  2000     0.1831320  0.5373833  0.12848911
##   1                  2500     0.1778574  0.5505235  0.12209313
##   1                  3000     0.1742092  0.5597227  0.11716936
##   1                  3500     0.1716621  0.5664189  0.11334564
##   1                  4000     0.1701625  0.5703639  0.11062390
##   1                  4500     0.1691255  0.5734536  0.10869947
##   1                  5000     0.1685429  0.5752649  0.10789840
##   1                  5500     0.1681944  0.5763153  0.10764586
##   1                  6000     0.1679193  0.5772218  0.10766962
##   1                  6500     0.1676691  0.5782582  0.10781953
##   1                  7000     0.1675026  0.5791145  0.10807441
##   1                  7500     0.1674431  0.5794730  0.10839219
##   1                  8000     0.1673553  0.5798363  0.10872500
##   1                  8500     0.1672696  0.5802706  0.10906576
##   1                  9000     0.1671723  0.5808824  0.10944028
##   1                  9500     0.1671933  0.5809542  0.10978272
##   1                 10000     0.1671539  0.5812307  0.11012546
##   1                 10500     0.1671547  0.5813962  0.11046063
##   1                 11000     0.1671482  0.5815500  0.11076698
##   1                 11500     0.1670776  0.5818552  0.11103891
##   1                 12000     0.1670899  0.5819566  0.11134365
##   1                 12500     0.1670744  0.5820467  0.11163698
##   1                 13000     0.1670629  0.5820531  0.11185851
##   1                 13500     0.1671021  0.5820431  0.11212352
##   1                 14000     0.1671688  0.5819046  0.11242023
##   1                 14500     0.1672609  0.5815236  0.11271889
##   1                 15000     0.1672414  0.5815486  0.11294315
##   1                 15500     0.1672694  0.5814595  0.11315290
##   1                 16000     0.1672855  0.5815873  0.11336020
##   1                 16500     0.1672617  0.5816539  0.11356570
##   1                 17000     0.1673041  0.5815828  0.11375661
##   1                 17500     0.1673724  0.5814112  0.11394510
##   1                 18000     0.1674393  0.5811127  0.11415674
##   1                 18500     0.1675226  0.5809154  0.11436165
##   1                 19000     0.1675932  0.5806968  0.11456401
##   1                 19500     0.1675779  0.5808066  0.11469208
##   1                 20000     0.1676236  0.5806323  0.11487004
##   1                 20500     0.1676828  0.5804456  0.11500101
##   1                 21000     0.1677423  0.5802403  0.11517986
##   1                 21500     0.1678338  0.5799192  0.11536466
##   1                 22000     0.1678938  0.5796839  0.11554429
##   1                 22500     0.1678971  0.5797352  0.11565831
##   1                 23000     0.1679281  0.5796376  0.11579512
##   1                 23500     0.1679813  0.5795677  0.11591496
##   1                 24000     0.1680311  0.5794820  0.11604862

```



##	1	24500	0.1680568	0.5792360	0.11615515
##	1	25000	0.1680816	0.5792260	0.11628105
##	1	25500	0.1681189	0.5791069	0.11638408
##	1	26000	0.1682174	0.5787820	0.11652927
##	1	26500	0.1682819	0.5786252	0.11663353
##	1	27000	0.1683325	0.5782847	0.11673454
##	1	27500	0.1683847	0.5781400	0.11682979
##	1	28000	0.1684494	0.5778956	0.11696772
##	1	28500	0.1685378	0.5776667	0.11708884
##	1	29000	0.1685966	0.5773497	0.11720647
##	1	29500	0.1686372	0.5771779	0.11731944
##	1	30000	0.1686736	0.5770251	0.11742131
##	1	30500	0.1687696	0.5767262	0.11751840
##	1	31000	0.1688152	0.5765563	0.11758834
##	1	31500	0.1688788	0.5763740	0.11768664
##	1	32000	0.1689261	0.5761759	0.11776235
##	1	32500	0.1689816	0.5760901	0.11785943
##	1	33000	0.1690188	0.5758197	0.11793775
##	1	33500	0.1690658	0.5757314	0.11801278
##	1	34000	0.1691419	0.5755570	0.11812002
##	1	34500	0.1691867	0.5754365	0.11818911
##	1	35000	0.1692360	0.5751593	0.11825524
##	1	35500	0.1692954	0.5749372	0.11836039
##	1	36000	0.1693476	0.5747394	0.11842810
##	1	36500	0.1694348	0.5744117	0.11853231
##	1	37000	0.1694931	0.5742627	0.11859583
##	1	37500	0.1695597	0.5741320	0.11868027
##	2	500	0.2137503	0.5604232	0.15977050
##	2	1000	0.1900088	0.5781564	0.13913312
##	2	1500	0.1765877	0.5921661	0.12559398
##	2	2000	0.1688755	0.6028566	0.11634537
##	2	2500	0.1643709	0.6105790	0.10959152
##	2	3000	0.1619216	0.6152882	0.10496616
##	2	3500	0.1606477	0.6185082	0.10192181
##	2	4000	0.1598136	0.6211583	0.09998822
##	2	4500	0.1593077	0.6231082	0.09880187
##	2	5000	0.1590381	0.6241381	0.09823526
##	2	5500	0.1589199	0.6245225	0.09810074
##	2	6000	0.1588604	0.6248718	0.09824476
##	2	6500	0.1589107	0.6249419	0.09854758
##	2	7000	0.1590157	0.6245981	0.09897750
##	2	7500	0.1590626	0.6243050	0.09928272
##	2	8000	0.1591175	0.6241695	0.09964640
##	2	8500	0.1592280	0.6238063	0.09997356
##	2	9000	0.1594571	0.6228850	0.10036185
##	2	9500	0.1596208	0.6221653	0.10070837
##	2	10000	0.1597952	0.6215747	0.10105302
##	2	10500	0.1600374	0.6206522	0.10140356
##	2	11000	0.1602157	0.6199648	0.10172315
##	2	11500	0.1604588	0.6191029	0.10203659
##	2	12000	0.1606167	0.6184754	0.10230569
##	2	12500	0.1608240	0.6176909	0.10255068
##	2	13000	0.1609856	0.6170830	0.10275346
##	2	13500	0.1610696	0.6165900	0.10293637

##	2	14000	0.1612863	0.6158137	0.10317954
##	2	14500	0.1614250	0.6152745	0.10330915
##	2	15000	0.1616299	0.6144505	0.10354205
##	2	15500	0.1617791	0.6139780	0.10371125
##	2	16000	0.1619087	0.6134491	0.10383206
##	2	16500	0.1621331	0.6126041	0.10402804
##	2	17000	0.1622810	0.6121171	0.10422085
##	2	17500	0.1624086	0.6117243	0.10436348
##	2	18000	0.1626084	0.6109255	0.10451729
##	2	18500	0.1627691	0.6103134	0.10465935
##	2	19000	0.1628951	0.6098289	0.10477137
##	2	19500	0.1631061	0.6090511	0.10497543
##	2	20000	0.1632228	0.6087294	0.10511400
##	2	20500	0.1633785	0.6080979	0.10522949
##	2	21000	0.1635344	0.6075315	0.10536598
##	2	21500	0.1636318	0.6071449	0.10544792
##	2	22000	0.1638008	0.6065304	0.10558707
##	2	22500	0.1639156	0.6060297	0.10570732
##	2	23000	0.1640671	0.6054805	0.10580877
##	2	23500	0.1641638	0.6051929	0.10589365
##	2	24000	0.1643228	0.6044657	0.10599951
##	2	24500	0.1644884	0.6037854	0.10613654
##	2	25000	0.1646500	0.6031625	0.10628593
##	2	25500	0.1647519	0.6026876	0.10637125
##	2	26000	0.1648764	0.6022171	0.10648756
##	2	26500	0.1649698	0.6017583	0.10656465
##	2	27000	0.1650527	0.6014732	0.10663942
##	2	27500	0.1651658	0.6010775	0.10675308
##	2	28000	0.1652546	0.6007621	0.10684296
##	2	28500	0.1653858	0.6001812	0.10696463
##	2	29000	0.1655380	0.5995648	0.10710268
##	2	29500	0.1656556	0.5990614	0.10718974
##	2	30000	0.1658216	0.5984188	0.10729817
##	2	30500	0.1659337	0.5979505	0.10738733
##	2	31000	0.1659881	0.5977160	0.10743805
##	2	31500	0.1660786	0.5974156	0.10753321
##	2	32000	0.1661791	0.5969972	0.10762031
##	2	32500	0.1663089	0.5963877	0.10774770
##	2	33000	0.1664091	0.5960499	0.10782755
##	2	33500	0.1665558	0.5954389	0.10793625
##	2	34000	0.1666956	0.5949787	0.10807434
##	2	34500	0.1668367	0.5943526	0.10819605
##	2	35000	0.1669858	0.5937913	0.10831560
##	2	35500	0.1670568	0.5935155	0.10836493
##	2	36000	0.1671539	0.5930449	0.10843982
##	2	36500	0.1672478	0.5926805	0.10853983
##	2	37000	0.1673684	0.5922293	0.10863388
##	2	37500	0.1674849	0.5917337	0.10872547
##	4	500	0.2081001	0.5930581	0.15506055
##	4	1000	0.1831339	0.6020570	0.13234731
##	4	1500	0.1705480	0.6106607	0.11842771
##	4	2000	0.1642763	0.6177361	0.10947892
##	4	2500	0.1611253	0.6229935	0.10379603
##	4	3000	0.1595433	0.6265126	0.10021965

##	4	3500	0.1588250	0.6284245	0.09816705
##	4	4000	0.1585888	0.6289045	0.09730911
##	4	4500	0.1586172	0.6289872	0.09712436
##	4	5000	0.1586710	0.6287279	0.09721216
##	4	5500	0.1588093	0.6281724	0.09753125
##	4	6000	0.1591624	0.6266933	0.09798206
##	4	6500	0.1594750	0.6252876	0.09844005
##	4	7000	0.1597877	0.6238037	0.09886474
##	4	7500	0.1601182	0.6224451	0.09933991
##	4	8000	0.1604669	0.6210524	0.09971809
##	4	8500	0.1608090	0.6195923	0.10008900
##	4	9000	0.1611486	0.6182290	0.10047200
##	4	9500	0.1614466	0.6169540	0.10078336
##	4	10000	0.1617722	0.6155133	0.10113906
##	4	10500	0.1620352	0.6144409	0.10145201
##	4	11000	0.1623149	0.6133813	0.10172519
##	4	11500	0.1626354	0.6119679	0.10201037
##	4	12000	0.1628977	0.6107972	0.10227710
##	4	12500	0.1631919	0.6096134	0.10252108
##	4	13000	0.1634325	0.6086692	0.10275018
##	4	13500	0.1636501	0.6077203	0.10292078
##	4	14000	0.1638746	0.6067647	0.10310736
##	4	14500	0.1640733	0.6058495	0.10329800
##	4	15000	0.1643461	0.6047095	0.10349918
##	4	15500	0.1645750	0.6037229	0.10366859
##	4	16000	0.1647998	0.6027834	0.10383637
##	4	16500	0.1649999	0.6018412	0.10400886
##	4	17000	0.1652350	0.6008822	0.10417221
##	4	17500	0.1654106	0.6001028	0.10435615
##	4	18000	0.1656218	0.5992080	0.10452839
##	4	18500	0.1658166	0.5982908	0.10468686
##	4	19000	0.1660728	0.5971898	0.10489168
##	4	19500	0.1662586	0.5964464	0.10507368
##	4	20000	0.1664480	0.5956697	0.10522151
##	4	20500	0.1666183	0.5950179	0.10535695
##	4	21000	0.1668123	0.5941862	0.10552806
##	4	21500	0.1669522	0.5936082	0.10564676
##	4	22000	0.1671218	0.5928233	0.10581134
##	4	22500	0.1672841	0.5921121	0.10596674
##	4	23000	0.1674383	0.5915153	0.10610127
##	4	23500	0.1676237	0.5907184	0.10626492
##	4	24000	0.1678090	0.5899697	0.10643673
##	4	24500	0.1679593	0.5892991	0.10658273
##	4	25000	0.1680940	0.5887025	0.10668507
##	4	25500	0.1683023	0.5878532	0.10685865
##	4	26000	0.1684177	0.5874002	0.10695632
##	4	26500	0.1685526	0.5868480	0.10707024
##	4	27000	0.1686666	0.5862840	0.10718260
##	4	27500	0.1688009	0.5856872	0.10729071
##	4	28000	0.1689137	0.5852355	0.10740345
##	4	28500	0.1690462	0.5846803	0.10755196
##	4	29000	0.1692570	0.5838181	0.10771089
##	4	29500	0.1693952	0.5832416	0.10782277
##	4	30000	0.1695108	0.5827616	0.10790706

##	4	30500	0.1696414	0.5823012	0.10801855
##	4	31000	0.1697798	0.5817395	0.10812054
##	4	31500	0.1698884	0.5812252	0.10823241
##	4	32000	0.1700139	0.5807376	0.10835756
##	4	32500	0.1701239	0.5802719	0.10844615
##	4	33000	0.1702166	0.5798486	0.10855412
##	4	33500	0.1703346	0.5794286	0.10864441
##	4	34000	0.1704742	0.5788458	0.10875689
##	4	34500	0.1705881	0.5782908	0.10885696
##	4	35000	0.1706793	0.5779450	0.10892784
##	4	35500	0.1707891	0.5774974	0.10900901
##	4	36000	0.1709009	0.5770026	0.10910501
##	4	36500	0.1710219	0.5765225	0.10918304
##	4	37000	0.1711095	0.5761821	0.10926702
##	4	37500	0.1712092	0.5757935	0.10936477
##	6	500	0.2062742	0.6012482	0.15339828
##	6	1000	0.1809392	0.6101613	0.12998589
##	6	1500	0.1687524	0.6164286	0.11614566
##	6	2000	0.1629721	0.6222686	0.10754441
##	6	2500	0.1603198	0.6261458	0.10229248
##	6	3000	0.1593016	0.6279180	0.09924148
##	6	3500	0.1588865	0.6288074	0.09777288
##	6	4000	0.1588843	0.6286181	0.09740707
##	6	4500	0.1591060	0.6275732	0.09748808
##	6	5000	0.1593639	0.6265285	0.09782845
##	6	5500	0.1597698	0.6246656	0.09834818
##	6	6000	0.1601350	0.6230271	0.09879440
##	6	6500	0.1605253	0.6213253	0.09918451
##	6	7000	0.1610039	0.6193616	0.09963463
##	6	7500	0.1614863	0.6172603	0.10010738
##	6	8000	0.1618985	0.6154619	0.10048556
##	6	8500	0.1622634	0.6139025	0.10082191
##	6	9000	0.1626910	0.6121194	0.10117574
##	6	9500	0.1630061	0.6106655	0.10146560
##	6	10000	0.1633947	0.6089960	0.10178402
##	6	10500	0.1637044	0.6076224	0.10205452
##	6	11000	0.1640731	0.6060814	0.10232065
##	6	11500	0.1644335	0.6045285	0.10261128
##	6	12000	0.1646876	0.6033523	0.10285240
##	6	12500	0.1649701	0.6021348	0.10312565
##	6	13000	0.1652288	0.6010556	0.10333379
##	6	13500	0.1654504	0.6000441	0.10352183
##	6	14000	0.1657430	0.5987300	0.10373237
##	6	14500	0.1660121	0.5975615	0.10395490
##	6	15000	0.1663250	0.5961947	0.10417761
##	6	15500	0.1665370	0.5952758	0.10431987
##	6	16000	0.1667689	0.5942024	0.10447840
##	6	16500	0.1669717	0.5932760	0.10463377
##	6	17000	0.1671798	0.5923788	0.10481005
##	6	17500	0.1673755	0.5916151	0.10496103
##	6	18000	0.1675964	0.5906946	0.10514607
##	6	18500	0.1678058	0.5898655	0.10532107
##	6	19000	0.1679845	0.5891321	0.10545738
##	6	19500	0.1681568	0.5882760	0.10564420

##	6	20000	0.1683034	0.5876082	0.10577413
##	6	20500	0.1684910	0.5868241	0.10595575
##	6	21000	0.1686735	0.5860131	0.10611852
##	6	21500	0.1688634	0.5851839	0.10628469
##	6	22000	0.1690340	0.5844878	0.10643509
##	6	22500	0.1691823	0.5839013	0.10657057
##	6	23000	0.1693301	0.5832654	0.10669796
##	6	23500	0.1694919	0.5825754	0.10683236
##	6	24000	0.1696317	0.5819980	0.10692455
##	6	24500	0.1697697	0.5814524	0.10704839
##	6	25000	0.1699091	0.5809223	0.10715336
##	6	25500	0.1700438	0.5802984	0.10728196
##	6	26000	0.1701859	0.5796818	0.10738237
##	6	26500	0.1703177	0.5791111	0.10749251
##	6	27000	0.1704677	0.5785403	0.10764465
##	6	27500	0.1705798	0.5780616	0.10773864
##	6	28000	0.1706827	0.5776539	0.10783402
##	6	28500	0.1707882	0.5772259	0.10790984
##	6	29000	0.1708922	0.5767972	0.10798989
##	6	29500	0.1710107	0.5762526	0.10809600
##	6	30000	0.1711197	0.5758264	0.10818352
##	6	30500	0.1712382	0.5753036	0.10829063
##	6	31000	0.1713437	0.5748381	0.10838585
##	6	31500	0.1714284	0.5744365	0.10846881
##	6	32000	0.1715349	0.5739525	0.10855328
##	6	32500	0.1716204	0.5736513	0.10862160
##	6	33000	0.1717187	0.5732653	0.10871153
##	6	33500	0.1718129	0.5729223	0.10879245
##	6	34000	0.1718861	0.5726156	0.10884123
##	6	34500	0.1719582	0.5723176	0.10889610
##	6	35000	0.1720431	0.5719629	0.10897737
##	6	35500	0.1721154	0.5716671	0.10902591
##	6	36000	0.1721900	0.5713887	0.10908647
##	6	36500	0.1722616	0.5710899	0.10914798
##	6	37000	0.1723336	0.5707940	0.10919233
##	6	37500	0.1723913	0.5705614	0.10923649
##	8	500	0.2056724	0.6043325	0.15266172
##	8	1000	0.1800684	0.6117354	0.12885505
##	8	1500	0.1682629	0.6170662	0.11515708
##	8	2000	0.1628897	0.6221450	0.10679539
##	8	2500	0.1604849	0.6255679	0.10180923
##	8	3000	0.1596672	0.6263910	0.09924485
##	8	3500	0.1594553	0.6264584	0.09825181
##	8	4000	0.1597797	0.6247565	0.09822361
##	8	4500	0.1601196	0.6230558	0.09851142
##	8	5000	0.1605655	0.6210311	0.09887941
##	8	5500	0.1610580	0.6191105	0.09927876
##	8	6000	0.1615487	0.6169906	0.09971674
##	8	6500	0.1620194	0.6149453	0.10012626
##	8	7000	0.1624932	0.6128172	0.10054394
##	8	7500	0.1629046	0.6109574	0.10083853
##	8	8000	0.1633469	0.6090814	0.10119174
##	8	8500	0.1637542	0.6072844	0.10149558
##	8	9000	0.1641326	0.6056509	0.10179599

##	8	9500	0.1645473	0.6038767	0.10209211
##	8	10000	0.1649385	0.6020996	0.10239276
##	8	10500	0.1652967	0.6005745	0.10267627
##	8	11000	0.1656526	0.5989190	0.10297880
##	8	11500	0.1659840	0.5975343	0.10324656
##	8	12000	0.1663013	0.5960288	0.10349512
##	8	12500	0.1666123	0.5946988	0.10367286
##	8	13000	0.1669615	0.5930681	0.10397115
##	8	13500	0.1672248	0.5919293	0.10417162
##	8	14000	0.1674720	0.5908541	0.10436238
##	8	14500	0.1677548	0.5895907	0.10455701
##	8	15000	0.1679971	0.5885377	0.10475239
##	8	15500	0.1682494	0.5874278	0.10493570
##	8	16000	0.1684328	0.5865559	0.10511171
##	8	16500	0.1686664	0.5855437	0.10532156
##	8	17000	0.1688702	0.5846732	0.10547201
##	8	17500	0.1690614	0.5838019	0.10564713
##	8	18000	0.1692411	0.5830555	0.10580668
##	8	18500	0.1694527	0.5821351	0.10598701
##	8	19000	0.1696136	0.5814892	0.10611391
##	8	19500	0.1697758	0.5807506	0.10624270
##	8	20000	0.1699712	0.5799023	0.10641804
##	8	20500	0.1701448	0.5791349	0.10657525
##	8	21000	0.1703082	0.5784205	0.10671977
##	8	21500	0.1704661	0.5778334	0.10684511
##	8	22000	0.1706072	0.5772613	0.10697183
##	8	22500	0.1707364	0.5767219	0.10708514
##	8	23000	0.1708670	0.5761375	0.10719049
##	8	23500	0.1709795	0.5756785	0.10727387
##	8	24000	0.1711308	0.5750210	0.10738757
##	8	24500	0.1712524	0.5745143	0.10749021
##	8	25000	0.1713637	0.5740602	0.10759146
##	8	25500	0.1714674	0.5736381	0.10767362
##	8	26000	0.1715815	0.5731544	0.10776158
##	8	26500	0.1716972	0.5726663	0.10784594
##	8	27000	0.1717854	0.5722727	0.10790558
##	8	27500	0.1718736	0.5719146	0.10798474
##	8	28000	0.1719573	0.5715688	0.10804256
##	8	28500	0.1720472	0.5711947	0.10811889
##	8	29000	0.1721249	0.5708551	0.10817994
##	8	29500	0.1722156	0.5704617	0.10824505
##	8	30000	0.1722912	0.5701667	0.10830692
##	8	30500	0.1723984	0.5697577	0.10837783
##	8	31000	0.1724673	0.5694767	0.10844187
##	8	31500	0.1725511	0.5691460	0.10851079
##	8	32000	0.1726409	0.5687683	0.10858323
##	8	32500	0.1726979	0.5685351	0.10862665
##	8	33000	0.1727582	0.5682779	0.10866226
##	8	33500	0.1728055	0.5681233	0.10870440
##	8	34000	0.1728673	0.5678887	0.10875261
##	8	34500	0.1729380	0.5675809	0.10881606
##	8	35000	0.1729945	0.5673634	0.10886621
##	8	35500	0.1730617	0.5670987	0.10891549
##	8	36000	0.1731237	0.5668235	0.10896268

##	8	36500	0.1731733	0.5666317	0.10899807
##	8	37000	0.1732122	0.5665012	0.10903885
##	8	37500	0.1732712	0.5662506	0.10909732
##	10	500	0.2050995	0.6070552	0.15213500
##	10	1000	0.1794826	0.6144966	0.12818444
##	10	1500	0.1678623	0.6190703	0.11473030
##	10	2000	0.1625576	0.6234839	0.10644799
##	10	2500	0.1604984	0.6254875	0.10181403
##	10	3000	0.1597459	0.6265611	0.09946826
##	10	3500	0.1596889	0.6260491	0.09869884
##	10	4000	0.1599152	0.6249949	0.09863401
##	10	4500	0.1602804	0.6235022	0.09885348
##	10	5000	0.1607582	0.6213569	0.09923379
##	10	5500	0.1612263	0.6191451	0.09956037
##	10	6000	0.1617702	0.6167579	0.09996534
##	10	6500	0.1622116	0.6147627	0.10033022
##	10	7000	0.1627188	0.6124314	0.10072645
##	10	7500	0.1632301	0.6101826	0.10110538
##	10	8000	0.1637088	0.6080121	0.10144854
##	10	8500	0.1641718	0.6060066	0.10176423
##	10	9000	0.1645616	0.6042373	0.10204506
##	10	9500	0.1650055	0.6022799	0.10234784
##	10	10000	0.1653673	0.6006051	0.10258157
##	10	10500	0.1656853	0.5991435	0.10284603
##	10	11000	0.1660691	0.5973076	0.10312015
##	10	11500	0.1663792	0.5959649	0.10332460
##	10	12000	0.1667406	0.5943463	0.10359089
##	10	12500	0.1670243	0.5931649	0.10378341
##	10	13000	0.1672967	0.5918803	0.10399682
##	10	13500	0.1675936	0.5906074	0.10421585
##	10	14000	0.1678361	0.5895497	0.10439079
##	10	14500	0.1680878	0.5884447	0.10459422
##	10	15000	0.1683000	0.5875320	0.10477279
##	10	15500	0.1685428	0.5865087	0.10496216
##	10	16000	0.1687551	0.5855014	0.10514819
##	10	16500	0.1689953	0.5844682	0.10533331
##	10	17000	0.1691620	0.5837118	0.10550409
##	10	17500	0.1693429	0.5829502	0.10565185
##	10	18000	0.1695334	0.5821340	0.10579576
##	10	18500	0.1696699	0.5815415	0.10591229
##	10	19000	0.1698174	0.5809213	0.10602152
##	10	19500	0.1700235	0.5800012	0.10616833
##	10	20000	0.1701643	0.5793453	0.10625569
##	10	20500	0.1703184	0.5786641	0.10636889
##	10	21000	0.1704776	0.5779786	0.10649549
##	10	21500	0.1706105	0.5774015	0.10660078
##	10	22000	0.1707360	0.5768998	0.10668013
##	10	22500	0.1708497	0.5764261	0.10676641
##	10	23000	0.1709721	0.5759802	0.10687519
##	10	23500	0.1710754	0.5755409	0.10697615
##	10	24000	0.1711622	0.5751896	0.10704606
##	10	24500	0.1712842	0.5746484	0.10713869
##	10	25000	0.1713792	0.5742640	0.10722314
##	10	25500	0.1714736	0.5738752	0.10729927

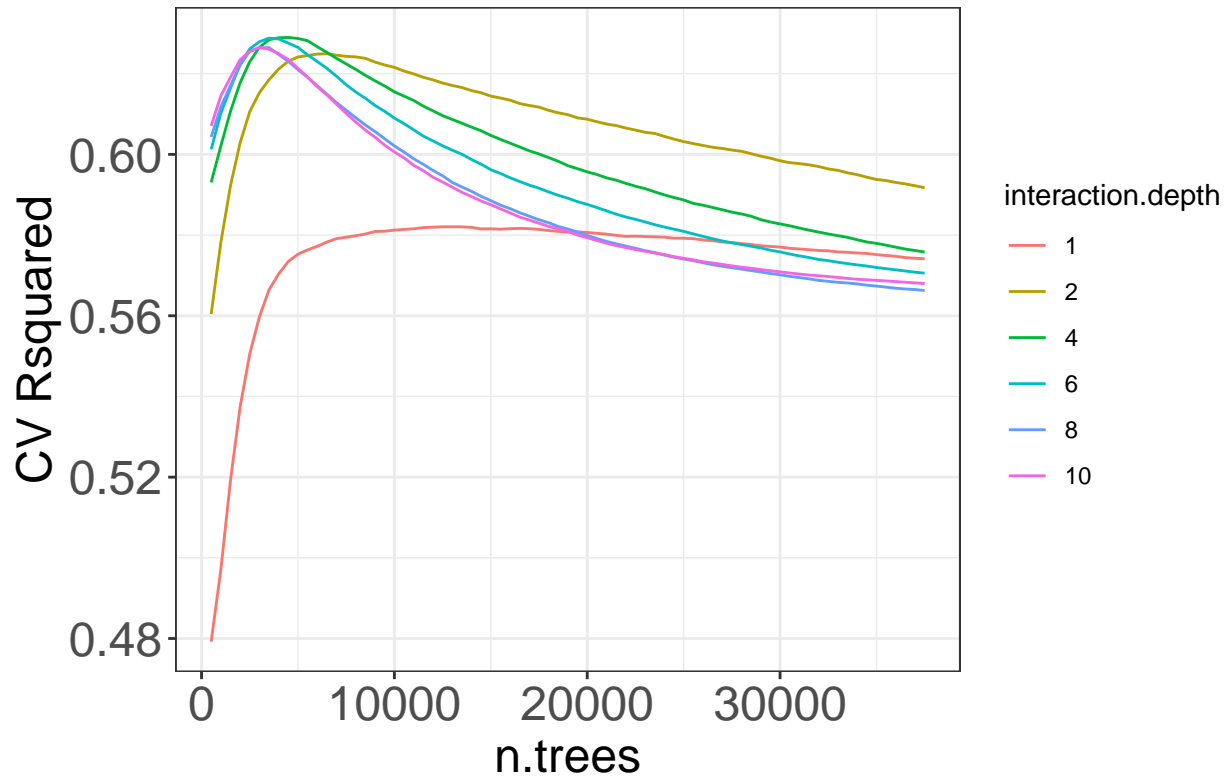
```
## 10      26000      0.1715899  0.5733928  0.10739968
## 10      26500      0.1716582  0.5730863  0.10745911
## 10      27000      0.1717587  0.5727277  0.10754360
## 10      27500      0.1718427  0.5723652  0.10760192
## 10      28000      0.1719220  0.5720314  0.10767169
## 10      28500      0.1719854  0.5717694  0.10772143
## 10      29000      0.1720531  0.5714521  0.10778694
## 10      29500      0.1721260  0.5711641  0.10785504
## 10      30000      0.1722062  0.5708877  0.10791577
## 10      30500      0.1722825  0.5705834  0.10797993
## 10      31000      0.1723429  0.5703322  0.10803090
## 10      31500      0.1724014  0.5700926  0.10808018
## 10      32000      0.1724527  0.5699104  0.10811601
## 10      32500      0.1725021  0.5697188  0.10816437
## 10      33000      0.1725510  0.5694976  0.10819819
## 10      33500      0.1726131  0.5692463  0.10825754
## 10      34000      0.1726633  0.5690446  0.10830037
## 10      34500      0.1726951  0.5689104  0.10832944
## 10      35000      0.1727387  0.5687660  0.10836671
## 10      35500      0.1727813  0.5686262  0.10840026
## 10      36000      0.1728243  0.5684690  0.10843440
## 10      36500      0.1728644  0.5683091  0.10847078
## 10      37000      0.1729034  0.5681476  0.10850440
## 10      37500      0.1729425  0.5679821  0.10854408
##
## Tuning parameter 'shrinkage' was held constant at a value of 0.001
##
## Tuning parameter 'n.minobsinnode' was held constant at a value of 10
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were n.trees = 4000,
## interaction.depth = 4, shrinkage = 0.001 and n.minobsinnode = 10.
```

```
best.boost <- train.boost$finalModel
pred.best.boost <- predict(best.boost, newdata = nba_csv.test, n.trees = 4000) # can use same model mat

ggplot(data = train.boost$results,
       mapping = aes(x = n.trees,
                     y = Rsquared,
                     colour = as.factor(interaction.depth))) +
  geom_line() +
  labs(y = "CV Rsquared", x = "n.trees",
       title = "Cross validating n.trees and interaction.depth (boosting)") +
  theme_bw() +
  theme(axis.title=element_text(size=18),
        axis.text=element_text(size=18),
        plot.title = element_text(size = 18)) +
  scale_color_discrete(name = "interaction.depth")
```



## Cross validating n.trees and interaction.depth (boo



```
# double-checking
# ggplot(train.boost$results,
#       aes(x = n.trees,
#           y = Rsquared,
#           colour = as.factor(interaction.depth))) +
#   geom_line() +
#   ylab("CV Rsquared") +
#   theme_bw() +
#   theme(axis.title=element_text(size=18),
#         axis.text=element_text(size=18)) +
#   scale_color_discrete(name = "interaction.depth")

SSE = sum((nba_csv.test$award_share - pred.best.boost)^2)
SST = sum((nba_csv.test$award_share - mean(nba_csv.train$award_share))^2)
OSR2 = 1 - SSE/SST
OSR2 #0.8231889
```

```
## [1] 0.8231889
```

## Bootstrapping for performance metrics

```
library(boot)
```

```

##
## Attaching package: 'boot'

## The following object is masked from 'package:car':
##
##      logit

## The following object is masked from 'package:lattice':
##
##      melanoma

boot_osr <- function(data, index) {
  labels <- data$label[index]
  predictions <- data$prediction[index]
  SSE <- sum((labels - predictions)^2)
  SST = sum((labels - mean_obs)^2)
  return(1 - SSE/SST)
}

boot_mae <- function(data, index) {
  labels <- data$label[index]
  predictions <- data$prediction[index]
  return(mean(abs(labels-predictions)))
}

boot_rmse <- function(data, index) {
  labels <- data$label[index]
  predictions <- data$prediction[index]
  return(sqrt(mean((labels-predictions)^2)))
}

boot_all_metrics <- function(data, index) {
  osr = boot_osr(data, index)
  mae = boot_mae(data, index)
  rmse = boot_rmse(data, index)
  return(c(osr, mae, rmse))
}

big_B = 10000

##baseline model
mean_obs <- mean(nba_csv.train$award_share) #0.1550612
predict.baseline = rep(mean_obs, nrow(nba_csv.test))
baseline_df = data.frame(labels = nba_csv.test$award_share, predictions = predict.baseline)
set.seed(6829)
Baseline_boot = boot(baseline_df, boot_all_metrics, R = big_B)
Baseline_boot

##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:

```

```
## boot(data = baseline_df, statistic = boot_all_metrics, R = big_B)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.0000000 0.0000000000 0.00000000
## t2* 0.2150102 -0.0004750286 0.06353995
## t3* 0.3090948 -0.0138000937 0.08911641
```

```
boot.ci(Baseline_boot, index = 1, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Baseline_boot, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%      ( 0, 0 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(Baseline_boot, index = 2, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Baseline_boot, type = "basic", index = 2)
##
## Intervals :
## Level      Basic
## 95%      ( 0.0776, 0.3208 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(Baseline_boot, index = 3, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Baseline_boot, type = "basic", index = 3)
##
## Intervals :
## Level      Basic
## 95%      ( 0.1625, 0.4977 )
## Calculations and Intervals on Original Scale
```

```
##naive lin reg
lin_df = data.frame(labels = nba_csv.test$award_share, predictions = predictions_testlm)
set.seed(342)
Lin_boot = boot(lin_df, boot_all_metrics, R = big_B)
Lin_boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = lin_df, statistic = boot_all_metrics, R = big_B)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.5684896 -1.400382e-01 0.38400668
## t2* 0.1772665  3.304536e-05 0.02856057
## t3* 0.2030427 -2.661079e-03 0.03412865
```

```
boot.ci(Lin_boot, index = 1, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Lin_boot, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%      ( 0.4050,  1.9182 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(Lin_boot, index = 2, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Lin_boot, type = "basic", index = 2)
##
## Intervals :
## Level      Basic
## 95%      ( 0.1165,  0.2289 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(Lin_boot, index = 3, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Lin_boot, type = "basic", index = 3)
##
## Intervals :
## Level      Basic
## 95%      ( 0.1363,  0.2635 )
## Calculations and Intervals on Original Scale
```

```
##stepwise lin reg
stepwise_df = data.frame(labels = nba_csv.test$award_share, predictions = pred.bswr)
set.seed(342)
Step_boot = boot(stepwise_df, boot_all_metrics, R = big_B)
Step_boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = stepwise_df, statistic = boot_all_metrics, R = big_B)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1*  0.5691312 -1.438696e-01  0.38844781
## t2*  0.1787127  3.674980e-06  0.02774371
## t3*  0.2028917 -2.379516e-03  0.03215793
```

```
boot.ci(Step_boot, index = 1, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Step_boot, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%      ( 0.4146,  1.9315 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(Step_boot, index = 2, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Step_boot, type = "basic", index = 2)
##
## Intervals :
## Level      Basic
## 95%      ( 0.1201,  0.2296 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(Step_boot, index = 3, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
```

```
## CALL :
## boot.ci(boot.out = Step_boot, type = "basic", index = 3)
##
## Intervals :
## Level      Basic
## 95%      ( 0.1397,  0.2602 )
## Calculations and Intervals on Original Scale

##random forest
rf_df = data.frame(labels = nba_csv.test$award_share, predictions = pred.best.rf)
set.seed(6722)
RF_boot = boot(rf_df, boot_all_metrics, R = big_B)
RF_boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = rf_df, statistic = boot_all_metrics, R = big_B)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.7677164 -1.048213e-01  0.28593204
## t2* 0.1248506  6.594466e-05  0.02327981
## t3* 0.1489708 -1.754376e-03  0.02357759
```

```
boot.ci(RF_boot, index = 1, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = RF_boot, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%      ( 0.6469,  1.7677 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(RF_boot, index = 2, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = RF_boot, type = "basic", index = 2)
##
## Intervals :
## Level      Basic
## 95%      ( 0.0775,  0.1690 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(RF_boot, index = 3, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = RF_boot, type = "basic", index = 3)
##
## Intervals :
## Level      Basic
## 95%      ( 0.1068,  0.1989 )
## Calculations and Intervals on Original Scale
```

```
##boosting
boost_df = data.frame(labels = nba_csv.test$award_share, predictions = pred.best.boost)
set.seed(9391)
Boost_boot = boot(boost_df, boot_all_metrics, R = big_B)
Boost_boot
```

```
##
## ORDINARY NONPARAMETRIC BOOTSTRAP
##
##
## Call:
## boot(data = boost_df, statistic = boot_all_metrics, R = big_B)
##
##
## Bootstrap Statistics :
##      original      bias    std. error
## t1* 0.82318889 -0.0897699945  0.27261546
## t2* 0.09692254  0.0002631934  0.02466090
## t3* 0.12997102 -0.0033825419  0.03044758
```

```
boot.ci(Boost_boot, index = 1, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Boost_boot, type = "basic", index = 1)
##
## Intervals :
## Level      Basic
## 95%      ( 0.7047,  1.8199 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(Boost_boot, index = 2, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
```

```
## CALL :
## boot.ci(boot.out = Boost_boot, type = "basic", index = 2)
##
## Intervals :
## Level      Basic
## 95%      ( 0.0434,  0.1399 )
## Calculations and Intervals on Original Scale
```

```
boot.ci(Boost_boot, index = 3, type = "basic")
```

```
## BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
## Based on 10000 bootstrap replicates
##
## CALL :
## boot.ci(boot.out = Boost_boot, type = "basic", index = 3)
##
## Intervals :
## Level      Basic
## 95%      ( 0.0746,  0.1925 )
## Calculations and Intervals on Original Scale
```

## Examining our predictions for MVP

```
all_preds <- filter(csv, season == "2018-19") %>%
  select(player, award_share)
# all_preds <- cbind(all_preds, pred.base, predictions_testlm,
#                   pred.bswr, pred.best.rf)
# colnames(all_preds) <- c("player", "award_share", "baseline",
#                          "linear", "bswr", "rf")
all_preds <- cbind(all_preds, pred.base, predictions_testlm,
                  pred.bswr, pred.best.rf, pred.best.boost)
colnames(all_preds) <- c("player", "award_share", "baseline",
                       "linear", "bswr", "rf", "boosting")
```



All\_preds dataframe. I forgot to call it during the knit() and then I didn't want to re-knit because the boosting cross-validation would take too long.

	player	award_share	baseline	linear	bswr	rf	boosting
1	Giannis Antetokounmpo	0.932	0.1550612	0.4918244	0.5039477	0.6606511	0.70751561
2	James Harden	0.768	0.1550612	0.5519403	0.5375057	0.6516786	0.84007466
3	Paul George	0.352	0.1550612	0.1647240	0.1740437	0.1220131	0.04379207
4	Nikola Jokić	0.210	0.1550612	0.3149604	0.3132550	0.2073275	0.12443941
5	Stephen Curry	0.173	0.1550612	0.1654079	0.1671717	0.2126188	0.03817124
6	Damian Lillard	0.068	0.1550612	0.2545112	0.2550683	0.1378830	0.05850163
7	Joel Embiid	0.049	0.1550612	0.1870386	0.1880855	0.1141124	0.06878112
8	Kevin Durant	0.025	0.1550612	0.2452853	0.2396639	0.1045116	0.03481834
9	Kawhi Leonard	0.013	0.1550612	0.1413281	0.1470319	0.1404159	0.05063030
10	Russell Westbrook	0.008	0.1550612	0.2404894	0.2419638	0.0990581	0.06817964
11	Rudy Gobert	0.001	0.1550612	0.1355675	0.1414766	0.2395502	0.12166169
12	LeBron James	0.001	0.1550612	0.1319142	0.1506761	0.1677270	0.08134474

## [IEOR 142] Scraper for NBA MVP Model

December 20, 2019

```
[4]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import requests
from bs4 import BeautifulSoup
import time
import re
```

```
[5]: from collections import defaultdict
```

```
[6]: #scrape player profiles
def work_player_profile(param, season):
    url = "https://www.basketball-reference.com" + param
    res = requests.get(url)
    soup = BeautifulSoup(res.text)

    data_dict = {}

    per_game = soup.find(attrs={'id': 'all_per_game'})
    for row in per_game.findAll("tr"):
        if 'id' in row.attrs and row.attrs['id'] == "per_game." + season:
            data_dict['fga'] = float(row.find('td', attrs={'data-stat': 'fga_per_g'}).text)
            data_dict['fg3a'] = float(row.find('td', attrs={'data-stat': 'fg3a_per_g'}).text)
            data_dict['fta'] = float(row.find('td', attrs={'data-stat': 'fta_per_g'}).text)
            break

    advanced_table = soup.find(attrs={'id': 'all_advanced'})
    for child in advanced_table.children:
        if "table_outer_container" in child:
            other_soup = BeautifulSoup(child)
            rows = other_soup.findAll("tr")
    for row in rows:
        if 'id' in row.attrs and row.attrs['id'] == "advanced." + season:
            data_dict.update(
```

```

        {
            'per': float(row.find('td', attrs={'data-stat': 'per'}).
↪text),
            'ts_pct': float(row.find('td', attrs={'data-stat': '
↪ts_pct'}).text),
            'usg_pct': float(row.find('td', attrs={'data-stat': '
↪usg_pct'}).text),
            'bpm': float(row.find('td', attrs={'data-stat': 'bpm'}).
↪text),
            'vorp': float(row.find('td', attrs={'data-stat': 'vorp'}).
↪text),
            'season': str(int(season)-1) + "-" + season[-2:],
        }
    )
    return data_dict

```

```

[7]: #scrape voting stats
def get_stats_of_voting(url):
    res = requests.get(url)
    soup = BeautifulSoup(res.text)
    item = soup.find(attrs={'class': 'stats_table'})
    rows = item.findAll("tr")

    season = url.split(".html")[0][-4:]

    print(f"Current season: {season}")

    players_stats = defaultdict(list)

    for index, row in enumerate(rows):

        print(f"\tCurrent index: {index} of {len(rows)}")
        header_cells = row.findAll("th")
        for header_cell in header_cells:
            if 'data-stat' in header_cell.attrs and header_cell['data-stat'] == '
↪ranker' and 'csk' in header_cell.attrs:
                rank = int(header_cell.getText())
            td_cells = row.findAll("td")
            if not td_cells:
                continue
            for cell in td_cells:
                if 'data-stat' not in cell.attrs:
                    continue
                # if cell['data-stat'] == 'age':
                #     continue
                if cell['data-stat'] == 'team_id':

```

```

base = "https://www.basketball-reference.com"
try:
    link = cell.find("a")['href']
except Exception:
    players_stats['win_pct'].append(0.5) # average
    continue
url = base + link
time.sleep(1)
soup = BeautifulSoup(requests.get(url).text)
for item in soup.findAll("p"):
    if "Record" in item.text:
        record = re.findall("\d+\-\d+", item.text)[0]
        splitted = record.split("-")
        players_stats['win_pct'].append(float(splitted[0]) /
↪(float(splitted[1]) + float(splitted[0])))
        break
    continue
if cell['data-stat'] == 'player':
    time.sleep(1)
    advanced_dict = work_player_profile(cell.find("a")['href'],
↪season)

    for key in advanced_dict:
        players_stats[key].append(advanced_dict[key])
    players_stats[cell['data-stat']].append(cell.getText())
else:
    text = cell.getText() or "0"
    players_stats[cell['data-stat']].append(float(text))
return players_stats

```

```

[10]: #setting csv parameters
seasons = range(1981, 2020)

new_data = defaultdict(list)

for season in seasons:
    full_url = f"https://www.basketball-reference.com/awards/
↪awards_{str(season)}.html"
    cur_season_dict = get_stats_of_voting(full_url)
    for key in cur_season_dict:
        new_data[key].extend(cur_season_dict[key])

data_frame = pd.DataFrame(new_data)

```

```

Current season: 1981
Current index: 0 of 33
Current index: 1 of 33
Current index: 2 of 33

```

Current index: 3 of 33  
Current index: 4 of 33  
Current index: 5 of 33  
Current index: 6 of 33  
Current index: 7 of 33  
Current index: 8 of 33  
Current index: 9 of 33  
Current index: 10 of 33  
Current index: 11 of 33  
Current index: 12 of 33  
Current index: 13 of 33  
Current index: 14 of 33  
Current index: 15 of 33  
Current index: 16 of 33  
Current index: 17 of 33  
Current index: 18 of 33  
Current index: 19 of 33  
Current index: 20 of 33  
Current index: 21 of 33  
Current index: 22 of 33  
Current index: 23 of 33  
Current index: 24 of 33  
Current index: 25 of 33  
Current index: 26 of 33  
Current index: 27 of 33  
Current index: 28 of 33  
Current index: 29 of 33  
Current index: 30 of 33  
Current index: 31 of 33  
Current index: 32 of 33  
Current season: 1982  
Current index: 0 of 27  
Current index: 1 of 27  
Current index: 2 of 27  
Current index: 3 of 27  
Current index: 4 of 27  
Current index: 5 of 27  
Current index: 6 of 27  
Current index: 7 of 27  
Current index: 8 of 27  
Current index: 9 of 27  
Current index: 10 of 27  
Current index: 11 of 27  
Current index: 12 of 27  
Current index: 13 of 27  
Current index: 14 of 27  
Current index: 15 of 27  
Current index: 16 of 27

Current index: 17 of 27  
Current index: 18 of 27  
Current index: 19 of 27  
Current index: 20 of 27  
Current index: 21 of 27  
Current index: 22 of 27  
Current index: 23 of 27  
Current index: 24 of 27  
Current index: 25 of 27  
Current index: 26 of 27  
Current season: 1983  
Current index: 0 of 25  
Current index: 1 of 25  
Current index: 2 of 25  
Current index: 3 of 25  
Current index: 4 of 25  
Current index: 5 of 25  
Current index: 6 of 25  
Current index: 7 of 25  
Current index: 8 of 25  
Current index: 9 of 25  
Current index: 10 of 25  
Current index: 11 of 25  
Current index: 12 of 25  
Current index: 13 of 25  
Current index: 14 of 25  
Current index: 15 of 25  
Current index: 16 of 25  
Current index: 17 of 25  
Current index: 18 of 25  
Current index: 19 of 25  
Current index: 20 of 25  
Current index: 21 of 25  
Current index: 22 of 25  
Current index: 23 of 25  
Current index: 24 of 25  
Current season: 1984  
Current index: 0 of 17  
Current index: 1 of 17  
Current index: 2 of 17  
Current index: 3 of 17  
Current index: 4 of 17  
Current index: 5 of 17  
Current index: 6 of 17  
Current index: 7 of 17  
Current index: 8 of 17  
Current index: 9 of 17  
Current index: 10 of 17

Current index: 11 of 17  
Current index: 12 of 17  
Current index: 13 of 17  
Current index: 14 of 17  
Current index: 15 of 17  
Current index: 16 of 17  
Current season: 1985  
Current index: 0 of 25  
Current index: 1 of 25  
Current index: 2 of 25  
Current index: 3 of 25  
Current index: 4 of 25  
Current index: 5 of 25  
Current index: 6 of 25  
Current index: 7 of 25  
Current index: 8 of 25  
Current index: 9 of 25  
Current index: 10 of 25  
Current index: 11 of 25  
Current index: 12 of 25  
Current index: 13 of 25  
Current index: 14 of 25  
Current index: 15 of 25  
Current index: 16 of 25  
Current index: 17 of 25  
Current index: 18 of 25  
Current index: 19 of 25  
Current index: 20 of 25  
Current index: 21 of 25  
Current index: 22 of 25  
Current index: 23 of 25  
Current index: 24 of 25  
Current season: 1986  
Current index: 0 of 20  
Current index: 1 of 20  
Current index: 2 of 20  
Current index: 3 of 20  
Current index: 4 of 20  
Current index: 5 of 20  
Current index: 6 of 20  
Current index: 7 of 20  
Current index: 8 of 20  
Current index: 9 of 20  
Current index: 10 of 20  
Current index: 11 of 20  
Current index: 12 of 20  
Current index: 13 of 20  
Current index: 14 of 20

Current index: 15 of 20  
Current index: 16 of 20  
Current index: 17 of 20  
Current index: 18 of 20  
Current index: 19 of 20  
Current season: 1987  
Current index: 0 of 19  
Current index: 1 of 19  
Current index: 2 of 19  
Current index: 3 of 19  
Current index: 4 of 19  
Current index: 5 of 19  
Current index: 6 of 19  
Current index: 7 of 19  
Current index: 8 of 19  
Current index: 9 of 19  
Current index: 10 of 19  
Current index: 11 of 19  
Current index: 12 of 19  
Current index: 13 of 19  
Current index: 14 of 19  
Current index: 15 of 19  
Current index: 16 of 19  
Current index: 17 of 19  
Current index: 18 of 19  
Current season: 1988  
Current index: 0 of 19  
Current index: 1 of 19  
Current index: 2 of 19  
Current index: 3 of 19  
Current index: 4 of 19  
Current index: 5 of 19  
Current index: 6 of 19  
Current index: 7 of 19  
Current index: 8 of 19  
Current index: 9 of 19  
Current index: 10 of 19  
Current index: 11 of 19  
Current index: 12 of 19  
Current index: 13 of 19  
Current index: 14 of 19  
Current index: 15 of 19  
Current index: 16 of 19  
Current index: 17 of 19  
Current index: 18 of 19  
Current season: 1989  
Current index: 0 of 21  
Current index: 1 of 21



Current index: 2 of 21  
Current index: 3 of 21  
Current index: 4 of 21  
Current index: 5 of 21  
Current index: 6 of 21  
Current index: 7 of 21  
Current index: 8 of 21  
Current index: 9 of 21  
Current index: 10 of 21  
Current index: 11 of 21  
Current index: 12 of 21  
Current index: 13 of 21  
Current index: 14 of 21  
Current index: 15 of 21  
Current index: 16 of 21  
Current index: 17 of 21  
Current index: 18 of 21  
Current index: 19 of 21  
Current index: 20 of 21  
Current season: 1990  
Current index: 0 of 16  
Current index: 1 of 16  
Current index: 2 of 16  
Current index: 3 of 16  
Current index: 4 of 16  
Current index: 5 of 16  
Current index: 6 of 16  
Current index: 7 of 16  
Current index: 8 of 16  
Current index: 9 of 16  
Current index: 10 of 16  
Current index: 11 of 16  
Current index: 12 of 16  
Current index: 13 of 16  
Current index: 14 of 16  
Current index: 15 of 16  
Current season: 1991  
Current index: 0 of 22  
Current index: 1 of 22  
Current index: 2 of 22  
Current index: 3 of 22  
Current index: 4 of 22  
Current index: 5 of 22  
Current index: 6 of 22  
Current index: 7 of 22  
Current index: 8 of 22  
Current index: 9 of 22  
Current index: 10 of 22

Current index: 11 of 22  
Current index: 12 of 22  
Current index: 13 of 22  
Current index: 14 of 22  
Current index: 15 of 22  
Current index: 16 of 22  
Current index: 17 of 22  
Current index: 18 of 22  
Current index: 19 of 22  
Current index: 20 of 22  
Current index: 21 of 22  
Current season: 1992  
Current index: 0 of 19  
Current index: 1 of 19  
Current index: 2 of 19  
Current index: 3 of 19  
Current index: 4 of 19  
Current index: 5 of 19  
Current index: 6 of 19  
Current index: 7 of 19  
Current index: 8 of 19  
Current index: 9 of 19  
Current index: 10 of 19  
Current index: 11 of 19  
Current index: 12 of 19  
Current index: 13 of 19  
Current index: 14 of 19  
Current index: 15 of 19  
Current index: 16 of 19  
Current index: 17 of 19  
Current index: 18 of 19  
Current season: 1993  
Current index: 0 of 16  
Current index: 1 of 16  
Current index: 2 of 16  
Current index: 3 of 16  
Current index: 4 of 16  
Current index: 5 of 16  
Current index: 6 of 16  
Current index: 7 of 16  
Current index: 8 of 16  
Current index: 9 of 16  
Current index: 10 of 16  
Current index: 11 of 16  
Current index: 12 of 16  
Current index: 13 of 16  
Current index: 14 of 16  
Current index: 15 of 16

Current season: 1994

Current index: 0 of 19  
Current index: 1 of 19  
Current index: 2 of 19  
Current index: 3 of 19  
Current index: 4 of 19  
Current index: 5 of 19  
Current index: 6 of 19  
Current index: 7 of 19  
Current index: 8 of 19  
Current index: 9 of 19  
Current index: 10 of 19  
Current index: 11 of 19  
Current index: 12 of 19  
Current index: 13 of 19  
Current index: 14 of 19  
Current index: 15 of 19  
Current index: 16 of 19  
Current index: 17 of 19  
Current index: 18 of 19

Current season: 1995

Current index: 0 of 17  
Current index: 1 of 17  
Current index: 2 of 17  
Current index: 3 of 17  
Current index: 4 of 17  
Current index: 5 of 17  
Current index: 6 of 17  
Current index: 7 of 17  
Current index: 8 of 17  
Current index: 9 of 17  
Current index: 10 of 17  
Current index: 11 of 17  
Current index: 12 of 17  
Current index: 13 of 17  
Current index: 14 of 17  
Current index: 15 of 17  
Current index: 16 of 17

Current season: 1996

Current index: 0 of 19  
Current index: 1 of 19  
Current index: 2 of 19  
Current index: 3 of 19  
Current index: 4 of 19  
Current index: 5 of 19  
Current index: 6 of 19  
Current index: 7 of 19  
Current index: 8 of 19

Current index: 9 of 19  
Current index: 10 of 19  
Current index: 11 of 19  
Current index: 12 of 19  
Current index: 13 of 19  
Current index: 14 of 19  
Current index: 15 of 19  
Current index: 16 of 19  
Current index: 17 of 19  
Current index: 18 of 19  
Current season: 1997  
Current index: 0 of 22  
Current index: 1 of 22  
Current index: 2 of 22  
Current index: 3 of 22  
Current index: 4 of 22  
Current index: 5 of 22  
Current index: 6 of 22  
Current index: 7 of 22  
Current index: 8 of 22  
Current index: 9 of 22  
Current index: 10 of 22  
Current index: 11 of 22  
Current index: 12 of 22  
Current index: 13 of 22  
Current index: 14 of 22  
Current index: 15 of 22  
Current index: 16 of 22  
Current index: 17 of 22  
Current index: 18 of 22  
Current index: 19 of 22  
Current index: 20 of 22  
Current index: 21 of 22  
Current season: 1998  
Current index: 0 of 21  
Current index: 1 of 21  
Current index: 2 of 21  
Current index: 3 of 21  
Current index: 4 of 21  
Current index: 5 of 21  
Current index: 6 of 21  
Current index: 7 of 21  
Current index: 8 of 21  
Current index: 9 of 21  
Current index: 10 of 21  
Current index: 11 of 21  
Current index: 12 of 21  
Current index: 13 of 21

Current index: 14 of 21  
Current index: 15 of 21  
Current index: 16 of 21  
Current index: 17 of 21  
Current index: 18 of 21  
Current index: 19 of 21  
Current index: 20 of 21  
Current season: 1999  
Current index: 0 of 23  
Current index: 1 of 23  
Current index: 2 of 23  
Current index: 3 of 23  
Current index: 4 of 23  
Current index: 5 of 23  
Current index: 6 of 23  
Current index: 7 of 23  
Current index: 8 of 23  
Current index: 9 of 23  
Current index: 10 of 23  
Current index: 11 of 23  
Current index: 12 of 23  
Current index: 13 of 23  
Current index: 14 of 23  
Current index: 15 of 23  
Current index: 16 of 23  
Current index: 17 of 23  
Current index: 18 of 23  
Current index: 19 of 23  
Current index: 20 of 23  
Current index: 21 of 23  
Current index: 22 of 23  
Current season: 2000  
Current index: 0 of 18  
Current index: 1 of 18  
Current index: 2 of 18  
Current index: 3 of 18  
Current index: 4 of 18  
Current index: 5 of 18  
Current index: 6 of 18  
Current index: 7 of 18  
Current index: 8 of 18  
Current index: 9 of 18  
Current index: 10 of 18  
Current index: 11 of 18  
Current index: 12 of 18  
Current index: 13 of 18  
Current index: 14 of 18  
Current index: 15 of 18

Current index: 16 of 18  
Current index: 17 of 18  
Current season: 2001  
Current index: 0 of 19  
Current index: 1 of 19  
Current index: 2 of 19  
Current index: 3 of 19  
Current index: 4 of 19  
Current index: 5 of 19  
Current index: 6 of 19  
Current index: 7 of 19  
Current index: 8 of 19  
Current index: 9 of 19  
Current index: 10 of 19  
Current index: 11 of 19  
Current index: 12 of 19  
Current index: 13 of 19  
Current index: 14 of 19  
Current index: 15 of 19  
Current index: 16 of 19  
Current index: 17 of 19  
Current index: 18 of 19  
Current season: 2002  
Current index: 0 of 20  
Current index: 1 of 20  
Current index: 2 of 20  
Current index: 3 of 20  
Current index: 4 of 20  
Current index: 5 of 20  
Current index: 6 of 20  
Current index: 7 of 20  
Current index: 8 of 20  
Current index: 9 of 20  
Current index: 10 of 20  
Current index: 11 of 20  
Current index: 12 of 20  
Current index: 13 of 20  
Current index: 14 of 20  
Current index: 15 of 20  
Current index: 16 of 20  
Current index: 17 of 20  
Current index: 18 of 20  
Current index: 19 of 20  
Current season: 2003  
Current index: 0 of 15  
Current index: 1 of 15  
Current index: 2 of 15  
Current index: 3 of 15

Current index: 4 of 15  
Current index: 5 of 15  
Current index: 6 of 15  
Current index: 7 of 15  
Current index: 8 of 15  
Current index: 9 of 15  
Current index: 10 of 15  
Current index: 11 of 15  
Current index: 12 of 15  
Current index: 13 of 15  
Current index: 14 of 15  
Current season: 2004  
Current index: 0 of 18  
Current index: 1 of 18  
Current index: 2 of 18  
Current index: 3 of 18  
Current index: 4 of 18  
Current index: 5 of 18  
Current index: 6 of 18  
Current index: 7 of 18  
Current index: 8 of 18  
Current index: 9 of 18  
Current index: 10 of 18  
Current index: 11 of 18  
Current index: 12 of 18  
Current index: 13 of 18  
Current index: 14 of 18  
Current index: 15 of 18  
Current index: 16 of 18  
Current index: 17 of 18  
Current season: 2005  
Current index: 0 of 18  
Current index: 1 of 18  
Current index: 2 of 18  
Current index: 3 of 18  
Current index: 4 of 18  
Current index: 5 of 18  
Current index: 6 of 18  
Current index: 7 of 18  
Current index: 8 of 18  
Current index: 9 of 18  
Current index: 10 of 18  
Current index: 11 of 18  
Current index: 12 of 18  
Current index: 13 of 18  
Current index: 14 of 18  
Current index: 15 of 18  
Current index: 16 of 18

Current index: 17 of 18  
Current season: 2006  
Current index: 0 of 13  
Current index: 1 of 13  
Current index: 2 of 13  
Current index: 3 of 13  
Current index: 4 of 13  
Current index: 5 of 13  
Current index: 6 of 13  
Current index: 7 of 13  
Current index: 8 of 13  
Current index: 9 of 13  
Current index: 10 of 13  
Current index: 11 of 13  
Current index: 12 of 13  
Current season: 2007  
Current index: 0 of 19  
Current index: 1 of 19  
Current index: 2 of 19  
Current index: 3 of 19  
Current index: 4 of 19  
Current index: 5 of 19  
Current index: 6 of 19  
Current index: 7 of 19  
Current index: 8 of 19  
Current index: 9 of 19  
Current index: 10 of 19  
Current index: 11 of 19  
Current index: 12 of 19  
Current index: 13 of 19  
Current index: 14 of 19  
Current index: 15 of 19  
Current index: 16 of 19  
Current index: 17 of 19  
Current index: 18 of 19  
Current season: 2008  
Current index: 0 of 19  
Current index: 1 of 19  
Current index: 2 of 19  
Current index: 3 of 19  
Current index: 4 of 19  
Current index: 5 of 19  
Current index: 6 of 19  
Current index: 7 of 19  
Current index: 8 of 19  
Current index: 9 of 19  
Current index: 10 of 19  
Current index: 11 of 19



Current index: 12 of 19  
Current index: 13 of 19  
Current index: 14 of 19  
Current index: 15 of 19  
Current index: 16 of 19  
Current index: 17 of 19  
Current index: 18 of 19  
Current season: 2009  
Current index: 0 of 14  
Current index: 1 of 14  
Current index: 2 of 14  
Current index: 3 of 14  
Current index: 4 of 14  
Current index: 5 of 14  
Current index: 6 of 14  
Current index: 7 of 14  
Current index: 8 of 14  
Current index: 9 of 14  
Current index: 10 of 14  
Current index: 11 of 14  
Current index: 12 of 14  
Current index: 13 of 14  
Current season: 2010  
Current index: 0 of 17  
Current index: 1 of 17  
Current index: 2 of 17  
Current index: 3 of 17  
Current index: 4 of 17  
Current index: 5 of 17  
Current index: 6 of 17  
Current index: 7 of 17  
Current index: 8 of 17  
Current index: 9 of 17  
Current index: 10 of 17  
Current index: 11 of 17  
Current index: 12 of 17  
Current index: 13 of 17  
Current index: 14 of 17  
Current index: 15 of 17  
Current index: 16 of 17  
Current season: 2011  
Current index: 0 of 15  
Current index: 1 of 15  
Current index: 2 of 15  
Current index: 3 of 15  
Current index: 4 of 15  
Current index: 5 of 15  
Current index: 6 of 15

Current index: 7 of 15  
Current index: 8 of 15  
Current index: 9 of 15  
Current index: 10 of 15  
Current index: 11 of 15  
Current index: 12 of 15  
Current index: 13 of 15  
Current index: 14 of 15  
Current season: 2012  
Current index: 0 of 17  
Current index: 1 of 17  
Current index: 2 of 17  
Current index: 3 of 17  
Current index: 4 of 17  
Current index: 5 of 17  
Current index: 6 of 17  
Current index: 7 of 17  
Current index: 8 of 17  
Current index: 9 of 17  
Current index: 10 of 17  
Current index: 11 of 17  
Current index: 12 of 17  
Current index: 13 of 17  
Current index: 14 of 17  
Current index: 15 of 17  
Current index: 16 of 17  
Current season: 2013  
Current index: 0 of 18  
Current index: 1 of 18  
Current index: 2 of 18  
Current index: 3 of 18  
Current index: 4 of 18  
Current index: 5 of 18  
Current index: 6 of 18  
Current index: 7 of 18  
Current index: 8 of 18  
Current index: 9 of 18  
Current index: 10 of 18  
Current index: 11 of 18  
Current index: 12 of 18  
Current index: 13 of 18  
Current index: 14 of 18  
Current index: 15 of 18  
Current index: 16 of 18  
Current index: 17 of 18  
Current season: 2014  
Current index: 0 of 19  
Current index: 1 of 19

Current index: 2 of 19  
Current index: 3 of 19  
Current index: 4 of 19  
Current index: 5 of 19  
Current index: 6 of 19  
Current index: 7 of 19  
Current index: 8 of 19  
Current index: 9 of 19  
Current index: 10 of 19  
Current index: 11 of 19  
Current index: 12 of 19  
Current index: 13 of 19  
Current index: 14 of 19  
Current index: 15 of 19  
Current index: 16 of 19  
Current index: 17 of 19  
Current index: 18 of 19  
Current season: 2015  
Current index: 0 of 14  
Current index: 1 of 14  
Current index: 2 of 14  
Current index: 3 of 14  
Current index: 4 of 14  
Current index: 5 of 14  
Current index: 6 of 14  
Current index: 7 of 14  
Current index: 8 of 14  
Current index: 9 of 14  
Current index: 10 of 14  
Current index: 11 of 14  
Current index: 12 of 14  
Current index: 13 of 14  
Current season: 2016  
Current index: 0 of 12  
Current index: 1 of 12  
Current index: 2 of 12  
Current index: 3 of 12  
Current index: 4 of 12  
Current index: 5 of 12  
Current index: 6 of 12  
Current index: 7 of 12  
Current index: 8 of 12  
Current index: 9 of 12  
Current index: 10 of 12  
Current index: 11 of 12  
Current season: 2017  
Current index: 0 of 13  
Current index: 1 of 13

```

Current index: 2 of 13
Current index: 3 of 13
Current index: 4 of 13
Current index: 5 of 13
Current index: 6 of 13
Current index: 7 of 13
Current index: 8 of 13
Current index: 9 of 13
Current index: 10 of 13
Current index: 11 of 13
Current index: 12 of 13
Current season: 2018
Current index: 0 of 15
Current index: 1 of 15
Current index: 2 of 15
Current index: 3 of 15
Current index: 4 of 15
Current index: 5 of 15
Current index: 6 of 15
Current index: 7 of 15
Current index: 8 of 15
Current index: 9 of 15
Current index: 10 of 15
Current index: 11 of 15
Current index: 12 of 15
Current index: 13 of 15
Current index: 14 of 15
Current season: 2019
Current index: 0 of 14
Current index: 1 of 14
Current index: 2 of 14
Current index: 3 of 14
Current index: 4 of 14
Current index: 5 of 14
Current index: 6 of 14
Current index: 7 of 14
Current index: 8 of 14
Current index: 9 of 14
Current index: 10 of 14
Current index: 11 of 14
Current index: 12 of 14
Current index: 13 of 14

```

```
[11]: data_frame
```

```

[11]:      fga  fg3a  fta  per  ts_pct  usg_pct  bpm  vorp  season  \
0    18.6   0.2   6.5  25.1   0.572    28.4   8.0   7.2  1980-81

```

1	18.3	0.9	4.0	19.9	0.528	24.3	5.1	5.8	1980-81
2	18.2	0.0	6.9	25.5	0.616	26.3	5.3	5.4	1980-81
3	19.3	0.0	10.1	25.1	0.585	27.6	3.7	4.7	1980-81
4	21.1	0.4	7.6	22.9	0.555	32.3	1.6	2.5	1980-81
5	15.2	0.1	5.0	22.0	0.583	23.3	5.9	5.1	1980-81
6	14.2	0.0	4.8	25.2	0.579	27.1	5.0	4.0	1980-81
7	15.4	0.6	6.3	17.0	0.516	24.6	2.0	2.6	1980-81
8	9.6	0.1	5.2	14.3	0.582	17.5	-1.3	0.5	1980-81
9	18.5	0.2	4.1	18.0	0.556	24.5	0.8	2.2	1980-81
10	15.9	0.5	6.1	25.7	0.582	24.3	9.2	3.9	1980-81
11	20.3	0.1	9.8	24.3	0.622	28.4	4.6	5.7	1980-81
12	13.4	0.5	5.4	17.3	0.553	23.3	-1.1	0.5	1980-81
13	15.4	0.1	5.4	19.9	0.617	23.2	3.3	3.9	1980-81
14	14.5	0.4	2.7	14.9	0.484	24.3	-2.8	-0.5	1980-81
15	16.0	0.1	5.0	18.0	0.512	23.6	1.5	2.6	1980-81
16	18.4	0.5	6.4	20.3	0.580	26.8	2.4	2.8	1980-81
17	14.1	1.3	4.3	17.1	0.511	20.0	4.5	5.2	1980-81
18	15.6	0.0	4.8	14.9	0.531	21.5	0.0	1.5	1980-81
19	13.4	1.1	6.7	18.4	0.578	23.1	2.1	3.0	1980-81
20	15.3	0.2	3.2	15.5	0.511	21.8	-1.2	0.6	1980-81
21	10.0	0.0	6.5	21.7	0.699	18.5	4.9	4.9	1980-81
22	10.7	0.0	4.1	19.7	0.573	21.8	3.4	2.4	1980-81
23	8.1	0.2	1.8	12.2	0.530	13.5	1.5	2.3	1980-81
24	14.1	0.2	3.2	18.4	0.579	25.6	0.9	1.6	1980-81
25	4.2	0.0	1.3	13.5	0.517	10.4	3.2	2.5	1980-81
26	9.3	0.0	4.3	20.1	0.604	20.8	5.0	3.6	1980-81
27	21.8	0.1	4.7	17.5	0.513	26.9	-1.5	0.4	1980-81
28	13.3	0.0	5.9	19.4	0.557	26.1	-0.3	0.9	1980-81
29	11.7	0.1	2.5	15.1	0.522	21.1	1.6	2.2	1980-81
..	...	...	...	...	...	...	...	...	
619	15.7	2.3	7.7	26.1	0.599	28.3	7.6	6.9	2016-17
620	18.4	3.5	6.8	23.2	0.541	30.6	4.1	4.3	2016-17
621	20.3	1.8	8.6	27.5	0.580	32.6	3.7	3.9	2016-17
622	16.5	5.0	6.2	27.6	0.651	27.8	8.0	5.2	2016-17
623	20.9	1.7	8.7	24.0	0.552	34.3	0.9	1.9	2016-17
624	20.1	10.0	10.1	29.8	0.619	36.1	10.9	8.3	2017-18
625	19.3	5.0	6.5	28.6	0.621	31.6	9.6	8.9	2017-18
626	19.5	2.2	8.0	28.9	0.612	30.0	5.2	4.9	2017-18
627	19.4	8.6	7.4	25.2	0.594	30.6	6.7	5.9	2017-18
628	21.1	4.1	7.1	24.7	0.524	34.1	8.2	7.5	2017-18
629	18.7	1.9	8.5	27.3	0.598	31.2	5.8	5.4	2017-18
630	18.0	6.1	5.9	26.0	0.640	30.4	5.6	4.5	2017-18
631	17.7	3.6	7.0	21.0	0.555	29.6	1.8	2.6	2017-18
632	18.0	1.2	5.3	25.0	0.570	29.1	3.3	3.3	2017-18
633	15.6	3.4	7.2	23.7	0.590	24.9	5.0	3.8	2017-18
634	16.9	9.8	5.9	28.2	0.675	31.0	8.6	4.4	2017-18
635	16.8	3.4	7.4	22.9	0.573	33.4	2.6	2.2	2017-18

636	17.9	5.8	4.9	23.1	0.577	30.1	4.9	4.5	2017-18
637	17.3	2.8	9.5	30.9	0.644	32.3	10.8	7.6	2018-19
638	24.5	13.2	11.0	30.6	0.616	40.5	11.7	9.9	2018-19
639	21.0	9.8	7.0	23.3	0.583	29.5	5.5	5.3	2018-19
640	15.1	3.4	4.4	26.3	0.589	27.4	9.5	7.3	2018-19
641	19.4	11.7	4.2	24.4	0.641	30.4	6.3	4.9	2018-19
642	19.2	8.0	6.4	23.7	0.588	29.3	5.5	5.4	2018-19
643	18.7	4.1	10.1	26.1	0.593	33.3	4.1	3.3	2018-19
644	17.7	5.0	6.5	24.2	0.631	29.0	4.3	4.3	2018-19
645	18.8	5.0	7.1	25.8	0.606	30.3	5.0	3.6	2018-19
646	20.2	5.6	6.2	21.1	0.501	30.9	6.5	5.6	2018-19
647	8.8	0.0	6.4	24.6	0.682	17.8	7.0	5.9	2018-19
648	19.9	5.9	7.6	25.6	0.588	31.6	8.1	4.9	2018-19

	player	...	pts_per_g	trb_per_g	ast_per_g	\
0	Julius Erving	...	24.6	8.0	4.4	
1	Larry Bird	...	21.2	10.9	5.5	
2	Kareem Abdul-Jabbar	...	26.2	10.3	3.4	
3	Moses Malone	...	27.8	14.8	1.8	
4	George Gervin	...	27.1	5.1	3.2	
5	Marques Johnson	...	20.3	6.8	4.6	
6	Robert Parish	...	18.9	9.5	1.8	
7	Dennis Johnson	...	18.8	4.6	3.7	
8	Tiny Archibald	...	13.8	2.2	7.7	
9	Jamaal Wilkes	...	22.6	5.4	2.9	
10	Magic Johnson	...	21.6	8.6	8.6	
11	Adrian Dantley	...	30.7	6.4	4.0	
12	Phil Ford	...	17.5	1.9	8.8	
13	Bernard King	...	21.9	6.8	3.5	
14	Kelvin Ransey	...	15.2	2.4	6.9	
15	Jack Sikma	...	18.7	10.4	3.0	
16	Otis Birdsong	...	24.6	3.6	3.3	
17	Micheal Ray Richardson	...	16.4	6.9	7.9	
18	Truck Robinson	...	18.8	9.6	2.5	
19	Reggie Theus	...	18.9	3.5	5.2	
20	Norm Nixon	...	17.1	2.9	8.8	
21	Artis Gilmore	...	17.9	10.1	2.1	
22	Bob Lanier	...	14.3	6.2	2.7	
23	Michael Cooper	...	9.4	4.1	4.1	
24	Walter Davis	...	18.0	2.6	3.9	
25	George Johnson	...	5.0	7.3	1.1	
26	Bobby Jones	...	13.5	5.4	2.8	
27	Mike Mitchell	...	24.5	6.1	1.7	
28	James Silas	...	17.7	3.1	3.8	
29	Quinn Buckner	...	13.3	3.6	4.7	
..	...	...	...	...	...	
619	Giannis Antetokounmpo	...	22.9	8.8	5.4	

620	John Wall	...	23.1	4.2	10.7
621	Anthony Davis	...	28.0	11.8	2.1
622	Kevin Durant	...	25.1	8.3	4.8
623	DeMar DeRozan	...	27.3	5.2	3.9
624	James Harden	...	30.4	5.4	8.8
625	LeBron James	...	27.5	8.6	9.1
626	Anthony Davis	...	28.1	11.1	2.3
627	Damian Lillard	...	26.9	4.5	6.6
628	Russell Westbrook	...	25.4	10.1	10.3
629	Giannis Antetokounmpo	...	26.9	10.0	4.8
630	Kevin Durant	...	26.4	6.8	5.4
631	DeMar DeRozan	...	23.0	3.9	5.2
632	LaMarcus Aldridge	...	23.1	8.5	2.0
633	Jimmy Butler	...	22.2	5.3	4.9
634	Stephen Curry	...	26.4	5.1	6.1
635	Joel Embiid	...	22.9	11.0	3.2
636	Victor Oladipo	...	23.1	5.2	4.3
637	Giannis Antetokounmpo	...	27.7	12.5	5.9
638	James Harden	...	36.1	6.6	7.5
639	Paul George	...	28.0	8.2	4.1
640	Nikola Jokić	...	20.1	10.8	7.3
641	Stephen Curry	...	27.3	5.3	5.2
642	Damian Lillard	...	25.8	4.6	6.9
643	Joel Embiid	...	27.5	13.6	3.7
644	Kevin Durant	...	26.0	6.4	5.9
645	Kawhi Leonard	...	26.6	7.3	3.3
646	Russell Westbrook	...	22.9	11.1	10.7
647	Rudy Gobert	...	15.9	12.9	2.0
648	LeBron James	...	27.4	8.5	8.3

	stl_per_g	blk_per_g	fg_pct	fg3_pct	ft_pct	ws	ws_per_48
0	2.1	1.8	0.521	0.222	0.787	13.8	0.231
1	2.0	0.8	0.478	0.270	0.863	10.8	0.160
2	0.7	2.9	0.574	0.000	0.766	14.3	0.230
3	1.0	1.9	0.522	0.333	0.757	13.7	0.202
4	1.1	0.7	0.492	0.257	0.826	10.5	0.182
5	1.5	0.5	0.552	0.000	0.706	11.2	0.211
6	1.0	2.6	0.545	0.000	0.710	10.9	0.228
7	1.7	0.8	0.436	0.216	0.820	8.4	0.154
8	0.9	0.2	0.499	0.000	0.816	6.9	0.118
9	1.5	0.4	0.526	0.077	0.758	8.5	0.135
10	3.4	0.7	0.532	0.176	0.760	6.4	0.225
11	1.4	0.2	0.559	0.286	0.806	13.6	0.191
12	1.5	0.1	0.478	0.306	0.831	5.2	0.110
13	0.9	0.4	0.588	0.333	0.703	9.1	0.150
14	1.1	0.1	0.452	0.097	0.749	2.8	0.056
15	1.0	1.1	0.454	0.000	0.823	8.0	0.132

16	1.3	0.3	0.544	0.286	0.697	8.2	0.152
17	2.9	0.4	0.469	0.225	0.663	6.9	0.104
18	0.8	0.5	0.505	0.000	0.629	7.6	0.119
19	1.5	0.2	0.495	0.200	0.809	9.2	0.156
20	1.8	0.1	0.476	0.167	0.778	5.5	0.089
21	0.6	2.4	0.670	0.000	0.705	12.3	0.208
22	1.1	1.2	0.525	1.000	0.751	6.8	0.185
23	1.6	1.0	0.491	0.211	0.785	5.0	0.092
24	1.2	0.2	0.539	0.412	0.836	7.1	0.157
25	0.6	3.4	0.473	0.000	0.734	4.5	0.112
26	1.2	0.9	0.539	0.000	0.813	9.2	0.217
27	0.8	0.6	0.476	0.444	0.784	6.0	0.091
28	0.7	0.2	0.477	0.000	0.850	7.0	0.164
29	2.4	0.0	0.493	0.167	0.734	5.4	0.108
..	...	...	...	...	...	...	...
619	1.6	1.9	0.521	0.272	0.770	12.4	0.210
620	2.0	0.6	0.451	0.327	0.801	8.8	0.149
621	1.3	2.2	0.505	0.299	0.802	11.0	0.195
622	1.1	1.6	0.537	0.375	0.875	12.0	0.278
623	1.1	0.2	0.467	0.266	0.842	9.0	0.166
624	1.8	0.7	0.449	0.367	0.858	15.4	0.289
625	1.4	0.9	0.542	0.367	0.731	14.0	0.221
626	1.5	2.6	0.534	0.340	0.828	13.7	0.241
627	1.1	0.4	0.439	0.361	0.916	12.6	0.227
628	1.8	0.3	0.449	0.298	0.737	10.1	0.166
629	1.5	1.4	0.529	0.307	0.760	11.9	0.207
630	0.7	1.8	0.516	0.419	0.889	10.4	0.215
631	1.1	0.3	0.456	0.310	0.825	9.6	0.170
632	0.6	1.2	0.510	0.293	0.837	10.9	0.209
633	2.0	0.4	0.474	0.350	0.854	8.9	0.198
634	1.6	0.2	0.495	0.423	0.921	9.1	0.267
635	0.6	1.8	0.483	0.308	0.769	6.2	0.155
636	2.4	0.8	0.477	0.371	0.799	8.2	0.155
637	1.3	1.5	0.578	0.256	0.729	14.4	0.292
638	2.0	0.7	0.442	0.368	0.879	15.2	0.254
639	2.2	0.4	0.438	0.386	0.839	11.9	0.201
640	1.4	0.7	0.511	0.307	0.821	11.8	0.226
641	1.3	0.4	0.472	0.437	0.916	9.7	0.199
642	1.1	0.4	0.444	0.369	0.912	12.1	0.205
643	0.7	1.9	0.484	0.300	0.804	8.7	0.194
644	0.7	1.1	0.521	0.353	0.885	11.5	0.204
645	1.8	0.4	0.496	0.371	0.854	9.5	0.224
646	1.9	0.5	0.428	0.290	0.656	6.8	0.124
647	0.8	2.3	0.669	0.000	0.636	14.4	0.268
648	1.3	0.6	0.510	0.339	0.665	7.2	0.179

[649 rows x 28 columns]



```
[12]: #make csv  
data_frame.to_csv("mvp_votings_1981-2020.csv")
```

```
[ ]:
```