Coursework_MAP501_2021

02/12/2021


```
library("car")
library("rio")
library("dplyr")
library("tidyr")
library("magrittr")
library("ggplot2")
library("pROC")
library("nnet")
library("caret")
library("lme4")
library("AmesHousing")
library("Lahman")
library("tidyverse")
library("here")
library("janitor")
library("readxl")
library("lindia")
library("glmnet")
library("caret")
library("lme4")
```

(1a)

Processing math: 100%

```
People %>%
  select(c(playerID, birthYear, nameFirst, nameLast, weight, height, bats, throws, debut, birth
Country)) %>%
  rename("bornUSA" = "birthCountry") %>%
  mutate(bornUSA = as.logical(as.factor(bornUSA) =="USA")) -> Peopledata
```

(1b)

```
Batting %>%
  filter(yearID == 1985 | yearID == 2015)  %>%
  select(!c(G, teamID, lgID)) %>%
  mutate(batav = case_when(H == 0 ~ 0, H > 0 ~ H/AB)) ->  Battingdata

Battingdata %>% sapply(function(x) sum(is.na(x)))

Fielding %>%
  filter(yearID == 1985 | yearID == 2015) %>%
  select(!c(G, teamID, lgID)) -> Fieldingdata1

Fieldingdata1 %>% sapply(function(x) sum(is.na(x)))

Fieldingdata1 %>%
  select(!c(PB, WP, SB, CS, ZR)) -> Fieldingdata

Fieldingdata
Battingdata
```

Processing math: 100%

```
playerID   yearID    stint       AB        R        H      X2B      X3B 
       0        0        0        0        0        0        0        0 
      HR      RBI       SB       CS       BB       SO      IBB      HBP 
       0        0        0        0        0        0        0        0 
      SH       SF     GIDP    batav 
       0        0        0        0 
playerID   yearID    stint      POS       GS  InnOuts       PO        A 
       0        0        0        0        0        0        0        0 
       E       DP       PB       WP       SB       CS       ZR 
       0        0     2995     3205     2995     2995     3205 
      playerID yearID stint POS  GS InnOuts  PO   A  E  DP
1     aasedo01   1985     1   P   0     264   8  10  0   0
2    abregjo01   1985     1   P   5      72   1   6  1   0
3    ackerji01   1985     1   P   0     259  10  16  0   1
4    adamsri02   1985     1  2B   3      84   9  13  1   1
5    adamsri02   1985     1  3B  10     337   2  31  1   3
6    adamsri02   1985     1  SS  19     476  24  57  3   9
7    agostju01   1985     1   P   0     181  10  15  1   0
8    aguaylu01   1985     1  2B   3     192  27  25  1   5
9    aguaylu01   1985     1  3B   3     126   4  16  0   1
10   aguaylu01   1985     1  SS  36    1052  61 117  8  21
11   aguilri01   1985     1   P  19     367   8  16  0   1
12   alexado01   1985     1   P  36     782  28  32  1   4
13   allenga01   1985     1   C  11     271  39   2  0   0
14   allenne01   1985     1   P   1      87   2   5  0   0
15   allenne01   1985     2   P   0      88   3   3  0   0
16   almonbi01   1985     1  1B   3      74  25   1  2   3
17   almonbi01   1985     1  3B   4     111   2   4  1   0
18   almonbi01   1985     1  OF  23     507  27   2  0   0
19   almonbi01   1985     1  SS  31     920  50 101  2  19
20   anderda02   1985     1  2B   1      36   4   2  0   1
21   anderda02   1985     1  3B  39    1100  28 107  6  10
22   anderda02   1985     1  SS  20     566  29  78  3   9
23   anderla02   1985     1   P   0     219   5  21  2   2
24   andujjo01   1985     1   P  38     809   8  45  6   8
25   armasto01   1985     1  OF  79    1962 173   3  3   1
26   armstmi01   1985     1   P   0      44   1   1  0   0
27   ashbyal01   1985     1   C  55    1414 312  37  8   1
28   atherke01   1985     1   P   0     314   4   6  1   0
29   ayalabe01   1985     1  OF  19     294  21   1  2   0
30   backmwa01   1985     1  2B 122    3384 272 370  7  76
31   backmwa01   1985     1  SS   0       6   1   0  0   0
32   bailema01   1985     1  1B   0       7   1   1  0   0
33   bailema01   1985     1   C  96    2651 565  51 13   6
34   bailobo01   1985     1  2B   6     209  18  30  0   5
35   bailobo01   1985     1  3B  18     642  14  63  3   6
36   bailobo01   1985     1  OF   0       1   0   0  0   0
37   bailobo01   1985     1  SS   2      75   3  10  0   1
38   baineha01   1985     1  OF 158    4193 318   8  2   2
39    bairdo01   1985     1   P   3     147   4   9  0   0
40    bairdo01   1985     2   P   0       6   0   0  0   0
41   bakerdo01   1985     1  2B   0       6   0   0  0   0
42   bakerdo01   1985     1  SS   6     177  12  12  1   2
43   bakerdu01   1985     1  1B  53    1258 400  26  3  33
44   bakerdu01   1985     1  OF  25     699  65   3  2   0
```

```
2474  0  1    8 0.27011494
2475  2  0    1 0.16883117
2476  0  0    0 0.00000000
2477  0  0    0 0.00000000
2478  6  0    0 0.15873016
2479  0 10   13 0.24855491
2480  0  0    0 0.00000000
2481  0  3    5 0.26808511
2482  0  2    3 0.28448276
2483  8  2    6 0.17428571
2484  0  0    0 0.00000000
```

(1c)

```
Salaries %>%
  filter(yearID == 1985 | yearID == 2015) %>%
  inner_join(Fieldingdata, by = c("yearID", "playerID"), keep = FALSE) %>%
  mutate(allstar = playerID %in% AllstarFull$playerID) %>%
  inner_join(Battingdata, by = c("yearID", "playerID"), keep = FALSE) %>%
  inner_join(Peopledata, by = c("playerID" = "playerID"), keep = FALSE) %>%
  mutate(age = yearID - birthYear) %>%
  rename(stint = stint.x) %>%
  drop_na() %>%
  droplevels() -> Playerdata

Playerdata
```

Processing math: 100%

	yearID	teamID	lgID	playerID	salary	stint	POS	GS	InnOuts	PO	A	E
1	1985	ATL	NL	barkele01	870000	1	P	18	221	2	9	1
2	1985	ATL	NL	bedrost01	550000	1	P	37	620	13	23	4
3	1985	ATL	NL	benedbr01	545000	1	C	67	1698	314	35	4
4	1985	ATL	NL	campri01	633333	1	P	2	383	7	13	4
5	1985	ATL	NL	ceronri01	625000	1	C	76	2097	384	48	6
6	1985	ATL	NL	chambch01	800000	1	1B	27	814	299	25	1
7	1985	ATL	NL	dedmoje01	150000	1	P	0	258	9	27	2
8	1985	ATL	NL	forstte01	483333	1	P	0	178	2	7	1
9	1985	ATL	NL	garbege01	772000	1	P	0	292	11	17	0
10	1985	ATL	NL	harpete01	250000	1	OF	124	3299	215	10	5
11	1985	ATL	NL	hornebo01	1500000	1	1B	85	2239	892	58	0
12	1985	ATL	NL	hornebo01	1500000	1	3B	40	957	25	61	11
13	1985	ATL	NL	hubbagl01	455000	1	2B	130	3425	339	539	10
14	1985	ATL	NL	mahleri01	407500	1	P	39	800	21	45	4
15	1985	ATL	NL	mcmurcr01	275000	1	P	6	135	2	12	2
16	1985	ATL	NL	mumphje01	775000	1	OF	113	3009	248	6	8
17	1985	ATL	NL	murphda05	1625000	1	OF	161	4264	334	8	7
18	1985	ATL	NL	oberkke01	616667	1	2B	12	275	18	37	1
19	1985	ATL	NL	oberkke01	616667	1	3B	101	2766	70	220	11
20	1985	ATL	NL	perezpa01	450000	1	P	22	286	7	9	1
21	1985	ATL	NL	perryge01	120000	1	1B	50	1319	541	37	9
22	1985	ATL	NL	perryge01	120000	1	OF	0	6	0	0	0
23	1985	ATL	NL	ramirra01	750000	1	SS	130	3466	214	451	32
24	1985	ATL	NL	suttebr01	1354167	1	P	0	265	5	13	0
25	1985	ATL	NL	washicl01	800000	1	OF	90	2459	122	3	5
26	1985	BAL	AL	boddimi01	625000	1	P	32	610	26	46	2
27	1985	BAL	AL	dauerri01	480000	1	1B	0	3	2	0	0
28	1985	BAL	AL	dauerri01	480000	1	2B	63	1504	117	181	3
29	1985	BAL	AL	dauerri01	480000	1	3B	8	250	7	21	1
30	1985	BAL	AL	davisst02	437500	1	P	28	525	15	20	0
31	1985	BAL	AL	dempsri01	512500	1	C	113	3024	575	49	8
32	1985	BAL	AL	dwyerji01	375000	1	OF	61	1494	131	4	1
33	1985	BAL	AL	flanami01	641667	1	P	15	258	4	11	0
34	1985	BAL	AL	grosswa01	483333	1	1B	5	132	40	4	0
35	1985	BAL	AL	grosswa01	483333	1	3B	57	1337	41	98	10
36	1985	BAL	AL	lacyle01	725000	1	OF	112	2946	231	9	4
37	1985	BAL	AL	lynnfr01	1090000	1	OF	121	3139	314	6	2
38	1985	BAL	AL	martide01	560000	1	P	31	540	17	26	1
39	1985	BAL	AL	martiti01	440000	1	P	0	210	9	10	1
40	1985	BAL	AL	mcgresc01	547143	1	P	34	612	13	26	1
41	1985	BAL	AL	murraed02	1472819	1	1B	154	4096	1338	152	19
42	1985	BAL	AL	nolanjo01	341667	1	C	4	97	22	2	0
43	1985	BAL	AL	rayfofl01	128500	1	3B	66	1829	62	145	6
44	1985	BAL	AL	rayfofl01	128500	1	C	22	575	114	7	1
45	1985	BAL	AL	ripkeca01	800000	1	SS	161	4282	286	474	26
46	1985	BAL	AL	roeniga01	558333	1	OF	53	1541	134	6	1
47	1985	BAL	AL	sheetla01	60000	1	1B	1	24	5	1	0
48	1985	BAL	AL	sheetla01	60000	1	OF	6	141	7	0	1
49	1985	BAL	AL	shelbjo01	130000	1	2B	0	3	0	1	0
50	1985	BAL	AL	shelbjo01	130000	1	OF	43	1262	148	3	3
51	1985	BAL	AL	stewasa01	581250	1	P	1	389	12	13	0
52	1985	BAL	AL	youngmi01	121000	1	OF	83	2239	190	6	5
53	1985	BOS	AL	armasto01	915000	1	OF	79	1962	173	3	3
54	1985	BOS	AL	barrema02	272500	1	2B	150	4007	355	479	11

Processing math: 100%

```
2330       L 2008-08-06    TRUE  30
2331       R 2012-04-28    TRUE  23
2332       R 2006-04-27    TRUE  34
2333       R 2003-04-17    TRUE  39
2334       R 2009-07-05   FALSE  31
2335       R 2012-04-29    TRUE  28
2336       R 2012-04-29    TRUE  28
2337       R 2012-04-29    TRUE  28
2338       R 2010-05-02   FALSE  28
2339       R 2013-04-21    TRUE  25
2340       R 2013-04-21    TRUE  25
2341       R 2013-08-07    TRUE  29
2342       L 2012-06-08    TRUE  30
2343       L 2012-06-08    TRUE  30
2344       L 2012-06-08    TRUE  30
2345       R 2008-04-29    TRUE  31
2346       L 2008-04-06    TRUE  31
2347       R 2009-05-21    TRUE  31
2348       R 2010-05-17    TRUE  28
2349       R 2010-06-08    TRUE  27
2350       R 2014-08-12    TRUE  24
2351       L 2004-06-27    TRUE  39
2352       R 2014-04-12    TRUE  27
2353       R 2006-04-03    TRUE  35
2354       R 2006-04-03    TRUE  35
2355       R 2002-09-01    TRUE  36
2356       R 2009-04-20    TRUE  29
2357       R 2005-09-01    TRUE  31
2358       R 2005-09-01    TRUE  31
```

(1d)

```
Salaries %>%
  group_by(teamID, yearID) %>%
  summarise(Rostercost = sum(salary), meansalary = mean(salary), rostersize = n_distinct(player
ID))  -> TeamSalaries

TeamSalaries
```

Processing math: 100%

```
# A tibble: 918 x 5
# Groups:   teamID [35]
   teamID yearID Rostercost meansalary rostersize
   <fct>   <int>      <int>      <dbl>      <int>
 1 ANA      1997   31135472   1004370.         31
 2 ANA      1998   41281000   1214147.         34
 3 ANA      1999   55388166   1384704.         40
 4 ANA      2000   51464167   1715472.         30
 5 ANA      2001   47535167   1584506.         30
 6 ANA      2002   61721667   2204345.         28
 7 ANA      2003   79031667   2927099.         27
 8 ANA      2004  100534667   3723506.         27
 9 ARI      1998   32347000    898528.         36
10 ARI      1999   68703999   2020706.         34
# ... with 908 more rows
```

(1e)

```
Teams %>%
  filter(yearID >= 1984, yearID <=2016) %>%
  inner_join(TeamSalaries, by = c("yearID", "teamID"), keep = FALSE) %>%
  drop_na() -> Teamdata

Teamdata
```

Processing math: 100%

	yearID	lgID	teamID	franchID	divID	Rank	G	Ghome	W	L	DivWin	WCWin	LgWin
1	1995	NL	ATL	ATL	E	1	144	72	90	54	Y	N	Y
2	1995	AL	BAL	BAL	E	3	144	72	71	73	N	N	N
3	1995	AL	BOS	BOS	E	1	144	72	86	58	Y	N	N
4	1995	AL	CAL	ANA	W	2	145	72	78	67	N	N	N
5	1995	AL	CHA	CHW	C	3	145	72	68	76	N	N	N
6	1995	NL	CHN	CHC	C	3	144	72	73	71	N	N	N
7	1995	NL	CIN	CIN	C	1	144	72	85	59	Y	N	N
8	1995	AL	CLE	CLE	C	1	144	72	100	44	Y	N	Y
9	1995	NL	COL	COL	W	2	144	72	77	67	N	Y	N
10	1995	AL	DET	DET	E	4	144	72	60	84	N	N	N
11	1995	NL	FLO	FLA	E	4	143	71	67	76	N	N	N
12	1995	NL	HOU	HOU	C	2	144	72	76	68	N	N	N
13	1995	AL	KCA	KCR	C	2	144	72	70	74	N	N	N
14	1995	NL	LAN	LAD	W	1	144	72	78	66	Y	N	N
15	1995	AL	MIN	MIN	C	5	144	72	56	88	N	N	N
16	1995	AL	ML4	MIL	C	4	144	72	65	79	N	N	N
17	1995	NL	MON	WSN	E	5	144	72	66	78	N	N	N
18	1995	AL	NYA	NYY	E	2	145	73	79	65	N	Y	N
19	1995	NL	NYN	NYM	E	2	144	72	69	75	N	N	N
20	1995	AL	OAK	OAK	W	4	144	72	67	77	N	N	N
21	1995	NL	PHI	PHI	E	2	144	72	69	75	N	N	N
22	1995	NL	PIT	PIT	C	5	144	72	58	86	N	N	N
23	1995	NL	SDN	SDP	W	3	144	72	70	74	N	N	N
24	1995	AL	SEA	SEA	W	1	145	73	79	66	Y	N	N
25	1995	NL	SFN	SFG	W	4	144	72	67	77	N	N	N
26	1995	NL	SLN	STL	C	4	143	72	62	81	N	N	N
27	1995	AL	TEX	TEX	W	3	144	72	74	70	N	N	N
28	1995	AL	TOR	TOR	E	5	144	72	56	88	N	N	N
29	1996	NL	ATL	ATL	E	1	162	81	96	66	Y	N	Y
30	1996	AL	BAL	BAL	E	2	163	82	88	74	N	Y	N
31	1996	AL	BOS	BOS	E	3	162	81	85	77	N	N	N
32	1996	AL	CAL	ANA	W	4	161	81	70	91	N	N	N
33	1996	AL	CHA	CHW	C	2	162	81	85	77	N	N	N
34	1996	NL	CHN	CHC	C	4	162	81	76	86	N	N	N
35	1996	NL	CIN	CIN	C	3	162	81	81	81	N	N	N
36	1996	AL	CLE	CLE	C	1	161	80	99	62	Y	N	N
37	1996	NL	COL	COL	W	3	162	81	83	79	N	N	N
38	1996	AL	DET	DET	E	5	162	81	53	109	N	N	N
39	1996	NL	FLO	FLA	E	3	162	81	80	82	N	N	N
40	1996	NL	HOU	HOU	C	2	162	81	82	80	N	N	N
41	1996	AL	KCA	KCR	C	5	161	80	75	86	N	N	N
42	1996	NL	LAN	LAD	W	2	162	81	90	72	N	Y	N
43	1996	AL	MIN	MIN	C	4	162	82	78	84	N	N	N
44	1996	AL	ML4	MIL	C	3	162	81	80	82	N	N	N
45	1996	NL	MON	WSN	E	2	162	81	88	74	N	N	N
46	1996	AL	NYA	NYY	E	1	162	80	92	70	Y	N	Y
47	1996	NL	NYN	NYM	E	4	162	81	71	91	N	N	N
48	1996	AL	OAK	OAK	W	3	162	81	78	84	N	N	N
49	1996	NL	PHI	PHI	E	5	162	81	67	95	N	N	N
50	1996	NL	PIT	PIT	C	5	162	80	73	89	N	N	N
51	1996	NL	SDN	SDP	W	1	162	81	91	71	Y	N	N
52	1996	AL	SEA	SEA	W	2	161	81	85	76	N	N	N
53	1996	NL	SFN	SFG	W	4	162	82	68	94	N	N	N
54	1996	NL	SLN	STL	C	1	162	81	88	74	Y	N	N

Processing math: 100%

```
627            BAL        BAL  161863456  5581498.5          29
628            BOS        BOS  188545761  6501578.0          29
629            CHA        CHA  112998667  4519946.7          25
630            CHN        CHN  154067668  5312678.2          29
631            CIN        CIN   88940059  3066898.6          29
632            CLE        CLE   74311900  2752292.6          27
633            COL        COL  112645071  3413487.0          33
634            DET        DET  194876481  6286338.1          31
635            HOU        HOU   94893700  3389060.7          28
636            KCA        KCA  131487125  4534038.8          29
637            ANA        ANA  137251333  5278897.4          26
638            LAN        LAN  221288380  6322525.1          35
639            FLO        MIA   77314202  2761221.5          28
640            ML4        MIL   68775237  2292507.9          30
641            MIN        MIN  102583200  4274300.0          24
642            NYA        NYA  222997792  7689579.0          29
643            NYN        NYN  133889129  4958856.6          27
644            OAK        OAK   86806234  2893541.1          30
645            PHI        PHI   58980000  2033793.1          29
646            PIT        PIT  103778833  3706386.9          28
647            SDN        SDN  101424814  3756474.6          27
648            SEA        SEA  135683339  4845833.5          28
649            SFN        SFN  172253778  6890151.1          25
650            SLN        SLN  143053500  4614629.0          31
651            TBA        TBA   57097310  2039189.6          28
652            TEX        TEX  176038723  6070300.8          29
653            TOR        TOR  138701700  4782817.2          29
654            MON        WAS  141652646  5448178.7          26
```

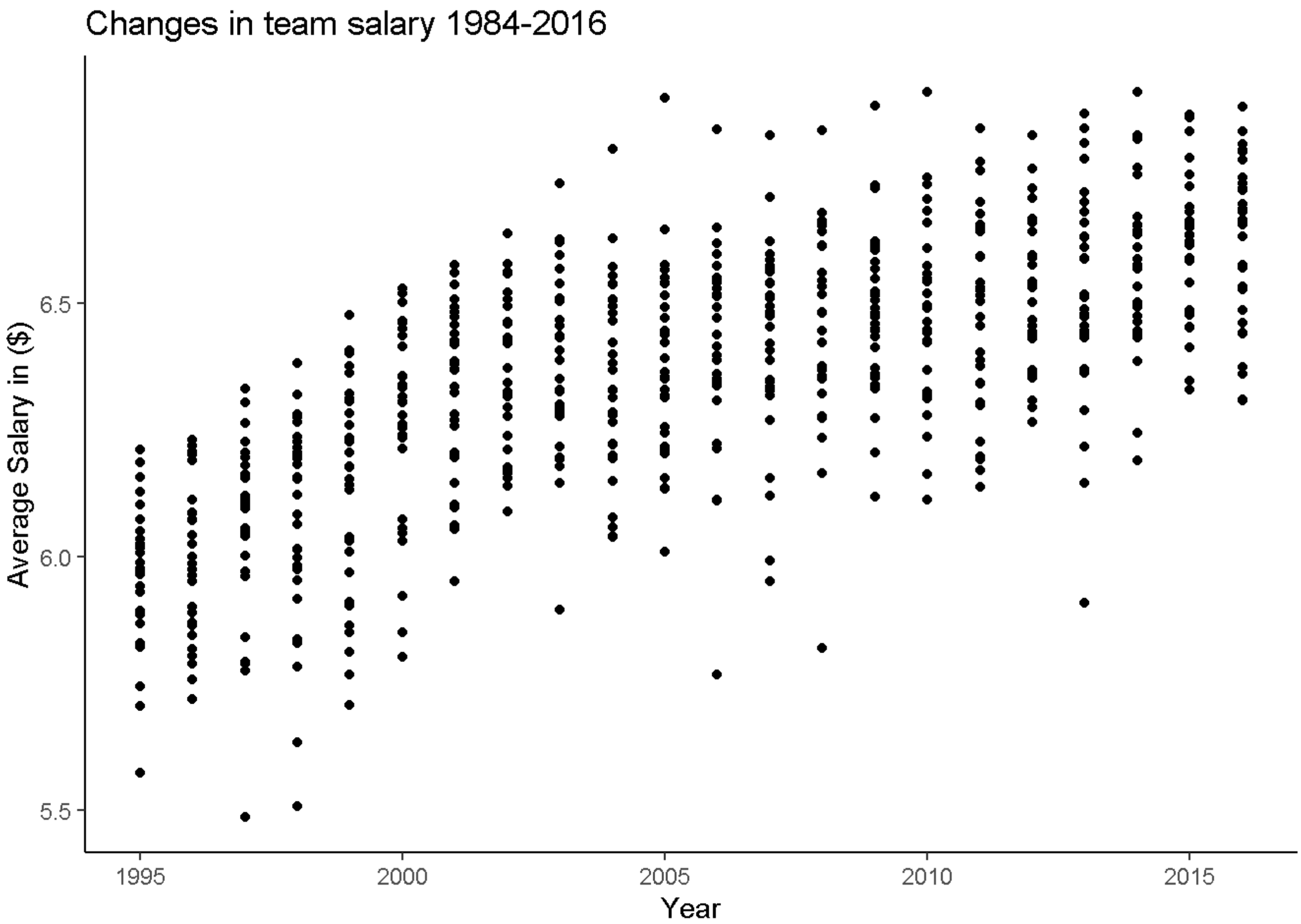
(2a)

```
# Regular Plot

Teamdata %>%
  ggplot(mapping = aes(x = yearID, y = meansalary)) +
  geom_point() +
  labs(x = "Year", y = "Average Salary in ($)") +
  ggtitle("Changes in team salary 1984-2016") +
  theme_classic()
```

Processing math: 100%


```
# Log Plot

Teamdata %>%
  ggplot(mapping = aes(x = yearID, y = log10(meansalary))) +
  geom_point() +
  labs(x = "Year", y = "Average Salary in ($)") +
  ggtitle("Changes in team salary 1984-2016") +
  theme_classic()
```

Processing math: 100%



Two reasons a linear model using log base 10 as opposed to the raw salary figures here could be:

- Using log 10 axis in a linear model allows for a cleaner visualization of the data which is easier to interpret and present to a stakeholder, the exponential values on an axis require are too difficult to quickly read and understand.
- Our log scale also allows us to plot values which are significantly higher/lower on the same chart without them warping the visualization. For example in the regular plot there were very low salaries in 95 and so it is difficult to see the spread of data down there when it’s on the same chart as the high salaries later on; the log scale also allows for outliar values such as those seen in 2005 not to warp the scale to such a significant degree.

(2b)

```
lm(log10(meansalary) ~ yearID, data = Teamdata) -> linmod1

linmod1

summary(linmod1)

linmod1$coefficients
```

Processing math: 100%

```
Call:
lm(formula = log10(meansalary) ~ yearID, data = Teamdata)

Coefficients:
(Intercept)       yearID
  -51.22242      0.02871


Call:
lm(formula = log10(meansalary) ~ yearID, data = Teamdata)

Residuals:
     Min       1Q   Median       3Q      Max
-0.66345 -0.11692  0.00644  0.13394  0.55976

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -51.222416   2.310867  -22.17   <2e-16 ***
yearID        0.028711   0.001152   24.92   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1858 on 652 degrees of freedom
Multiple R-squared:  0.4878,    Adjusted R-squared:  0.487
F-statistic: 620.9 on 1 and 652 DF,  p-value: < 2.2e-16

 (Intercept)       yearID
-51.22241645   0.02871141
```

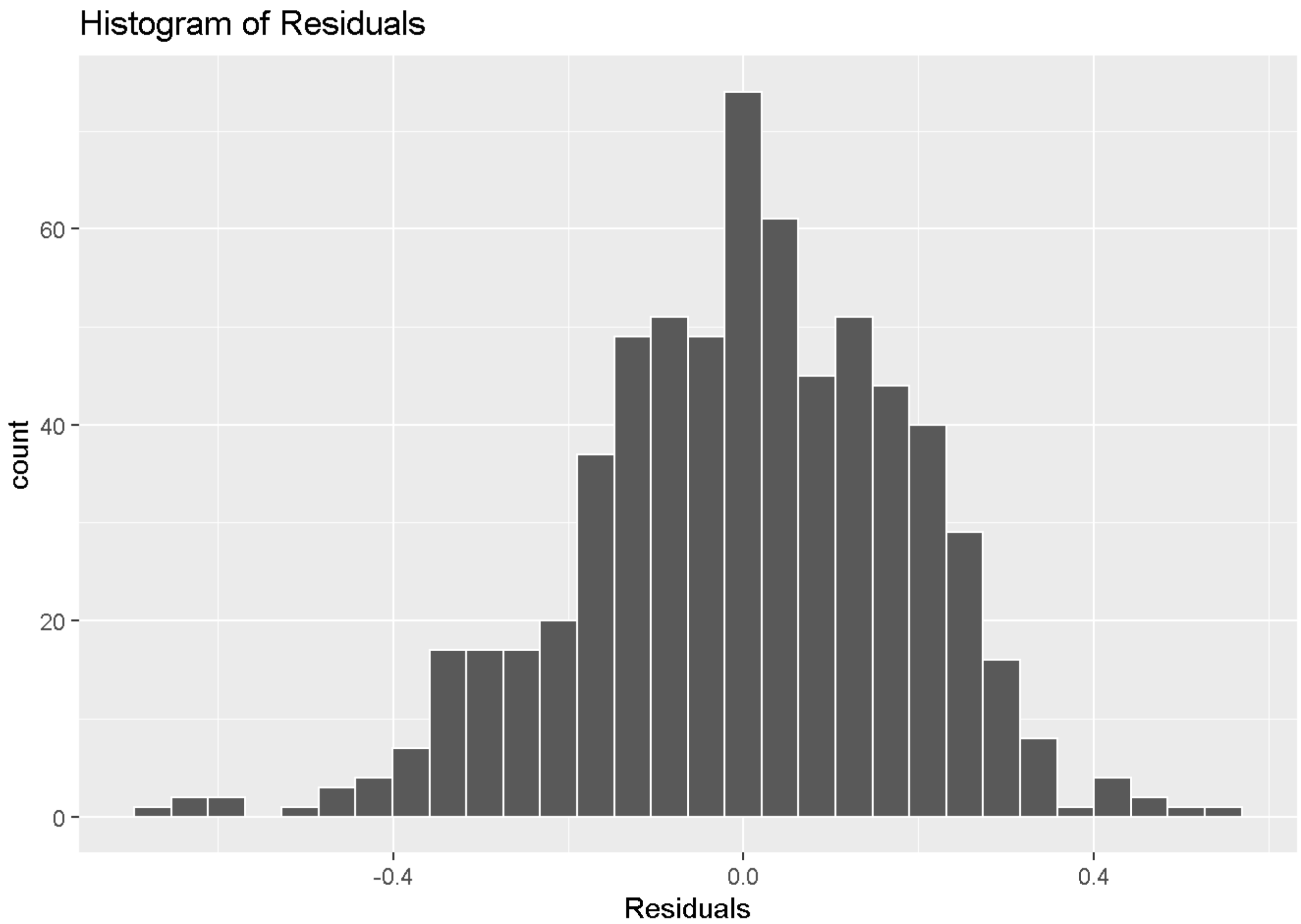
$$\log10(\text{meansalary}) = -51.22242 + 0.02871 \times \text{Year}$$

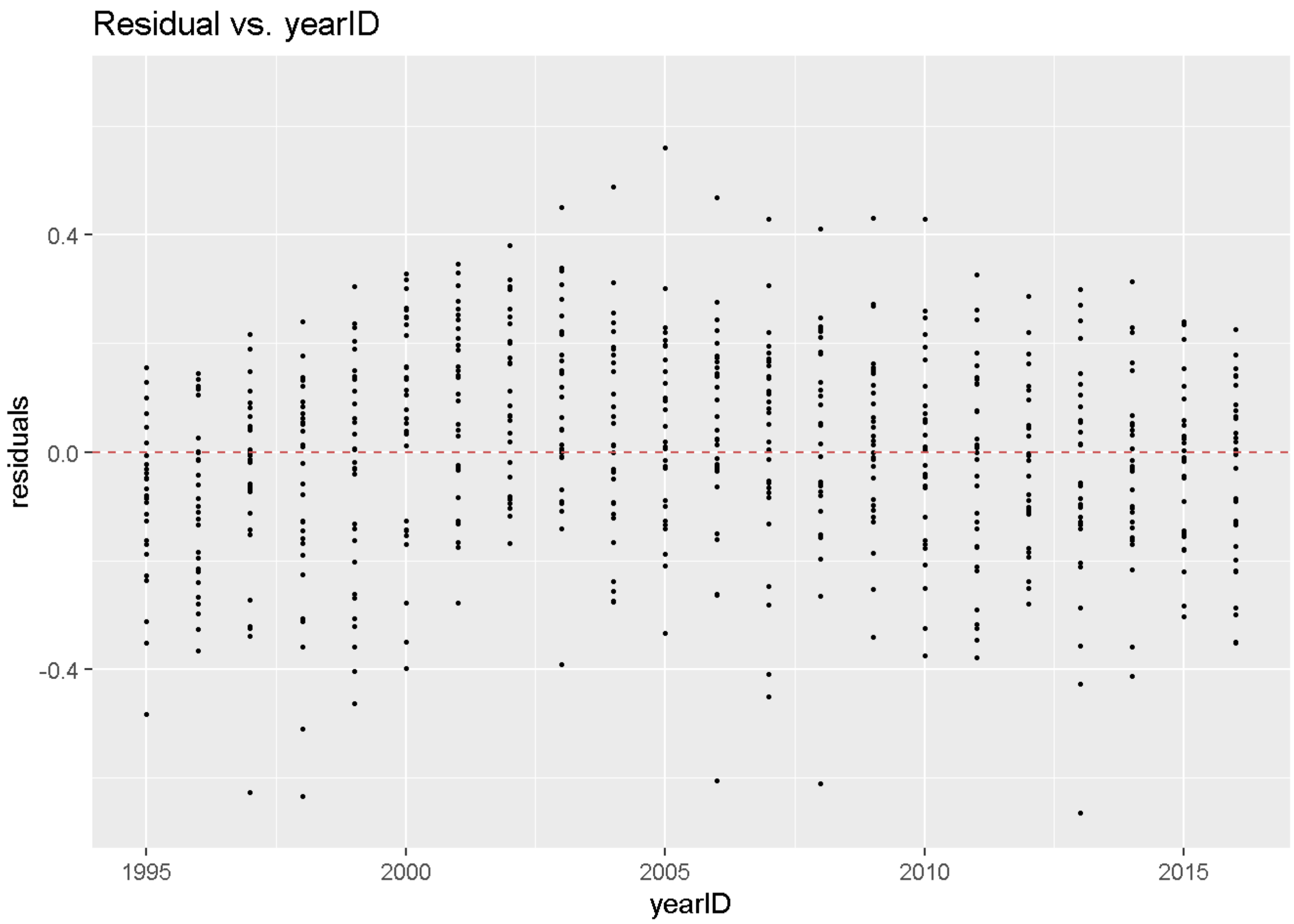
The multiple R squared here tells us that 48.78% of the variance seen in the log10(meansalary) value can be explained by the current year.

(2c)

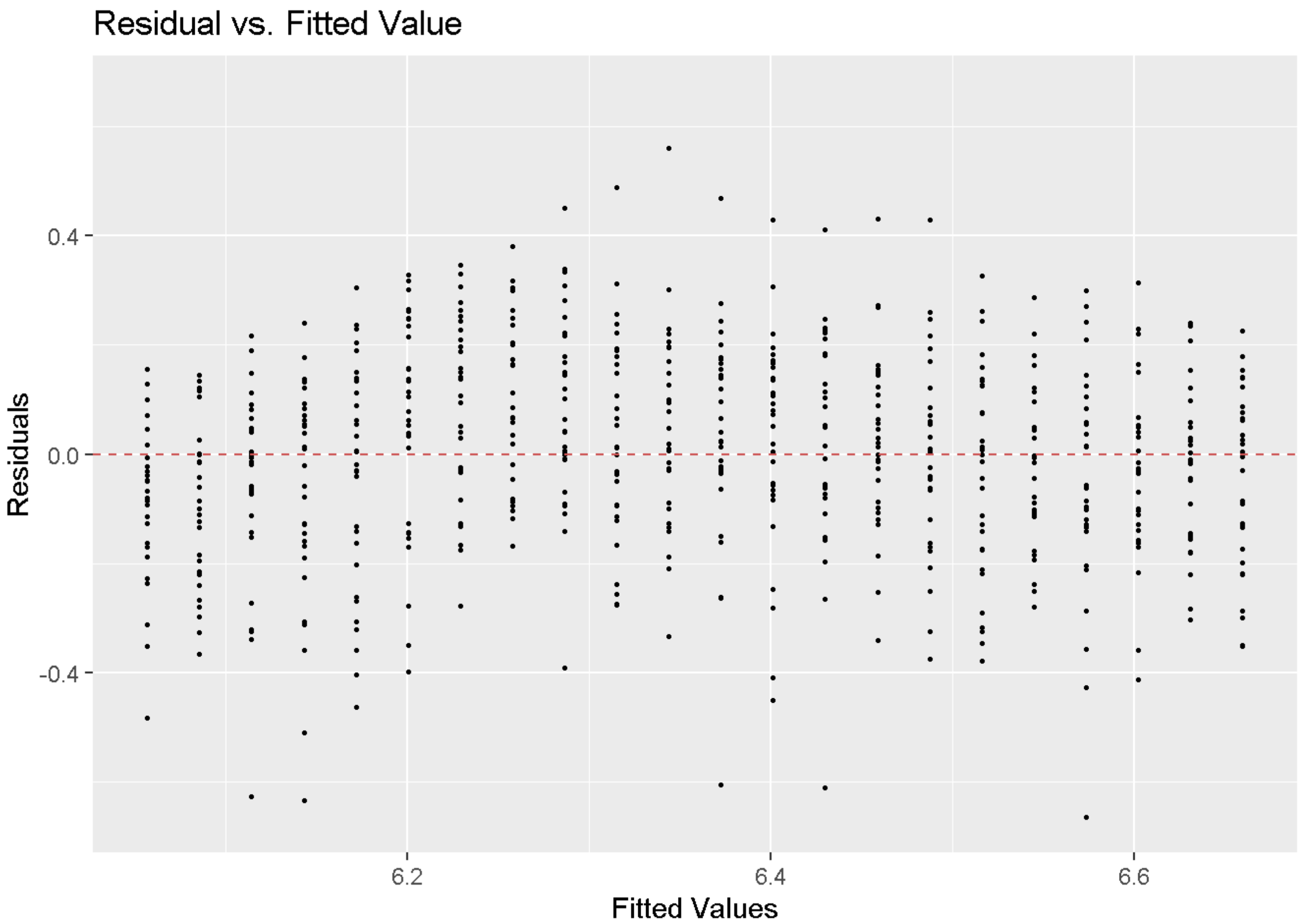
```
linmod1 %>%
  gg_diagnose(max.per.page = 1)
```

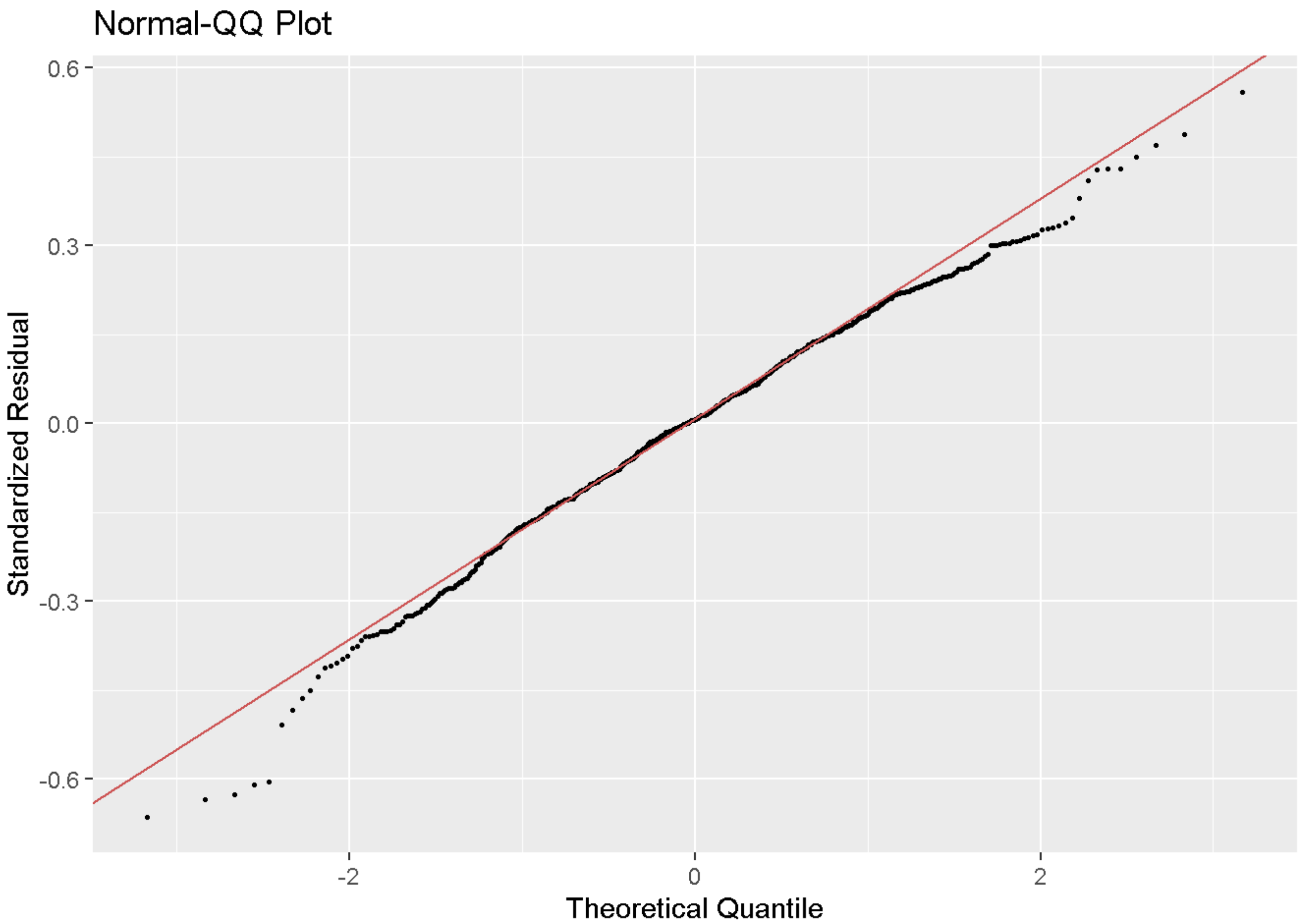
Processing math: 100%



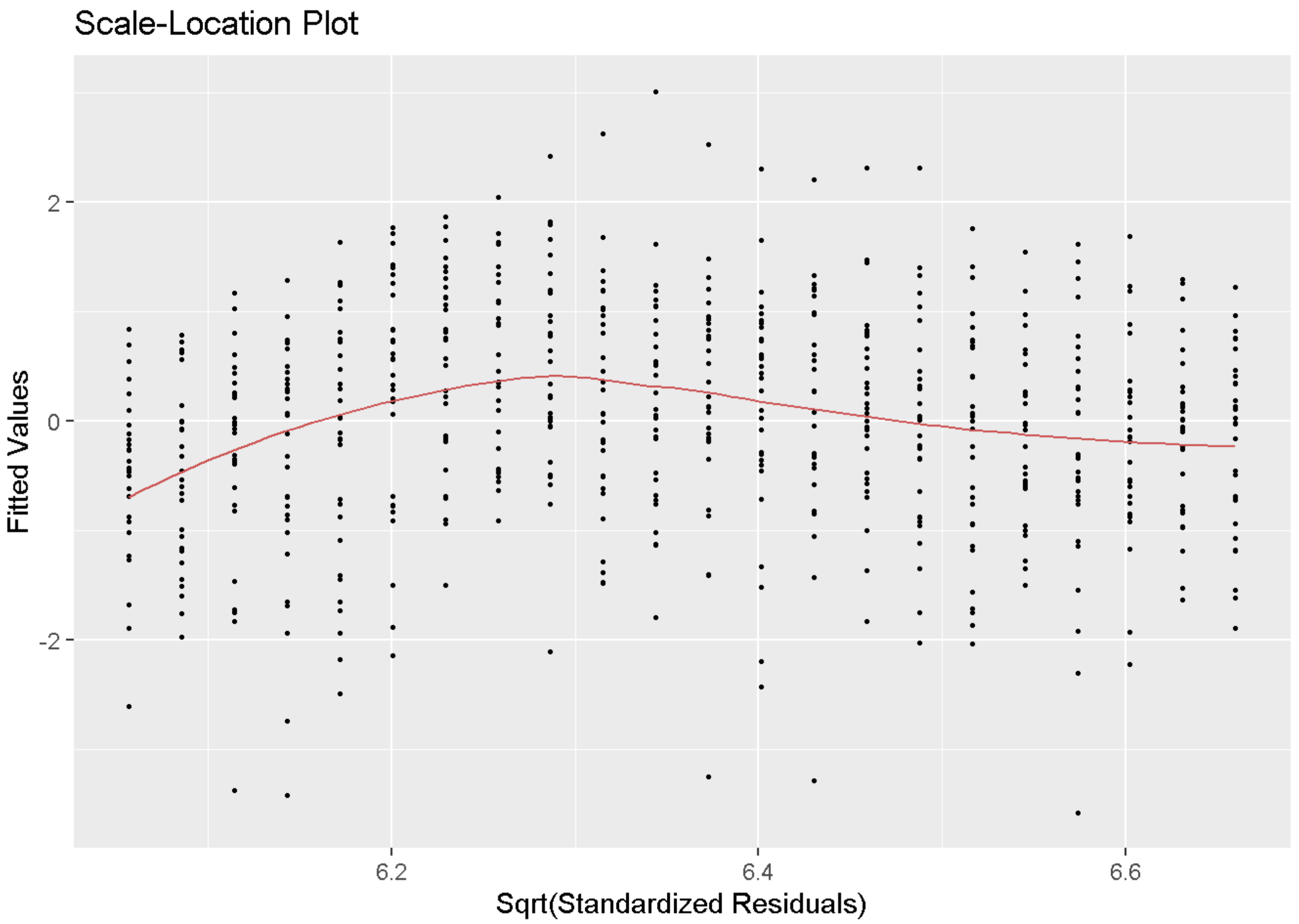


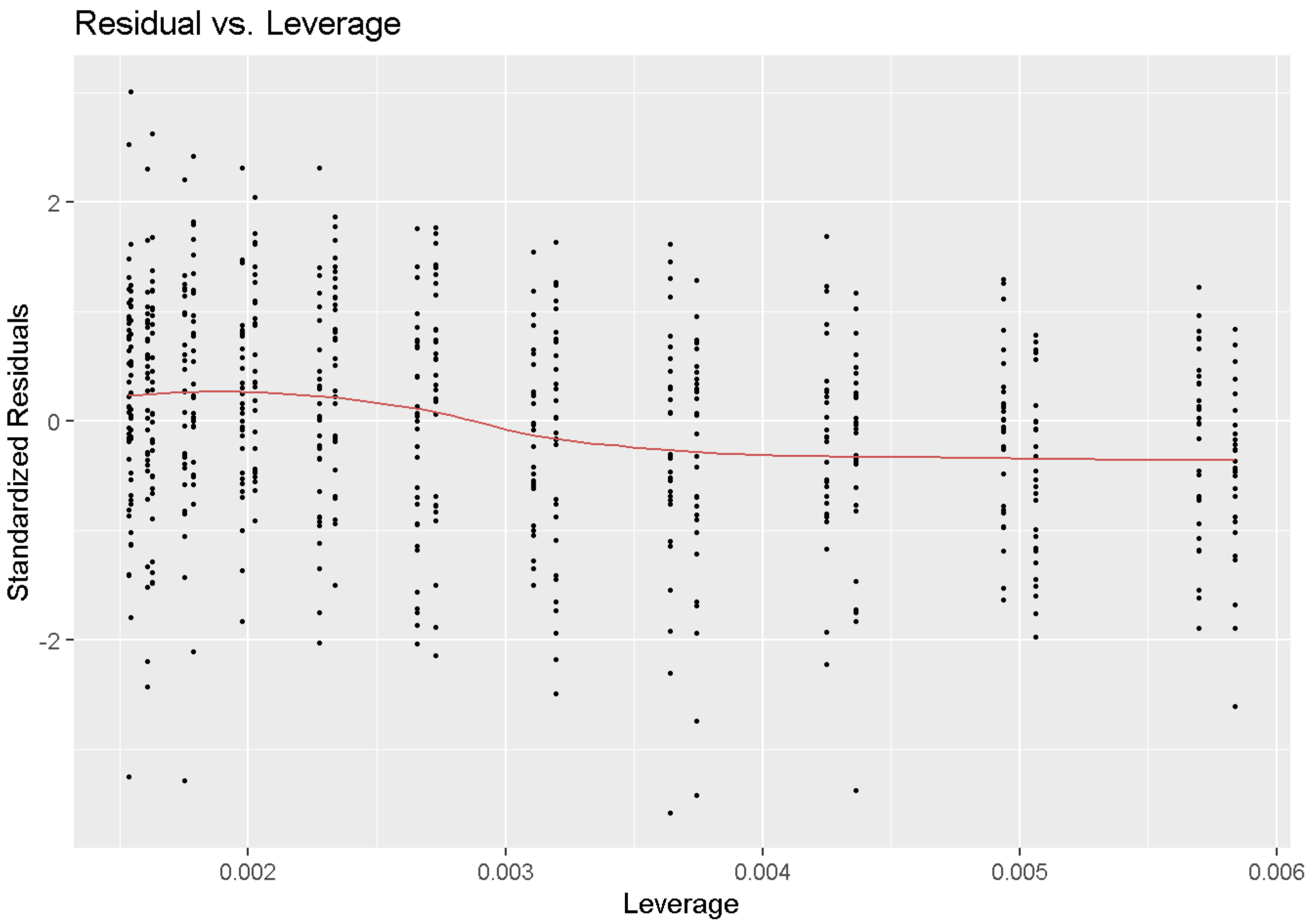
Processing math: 100%



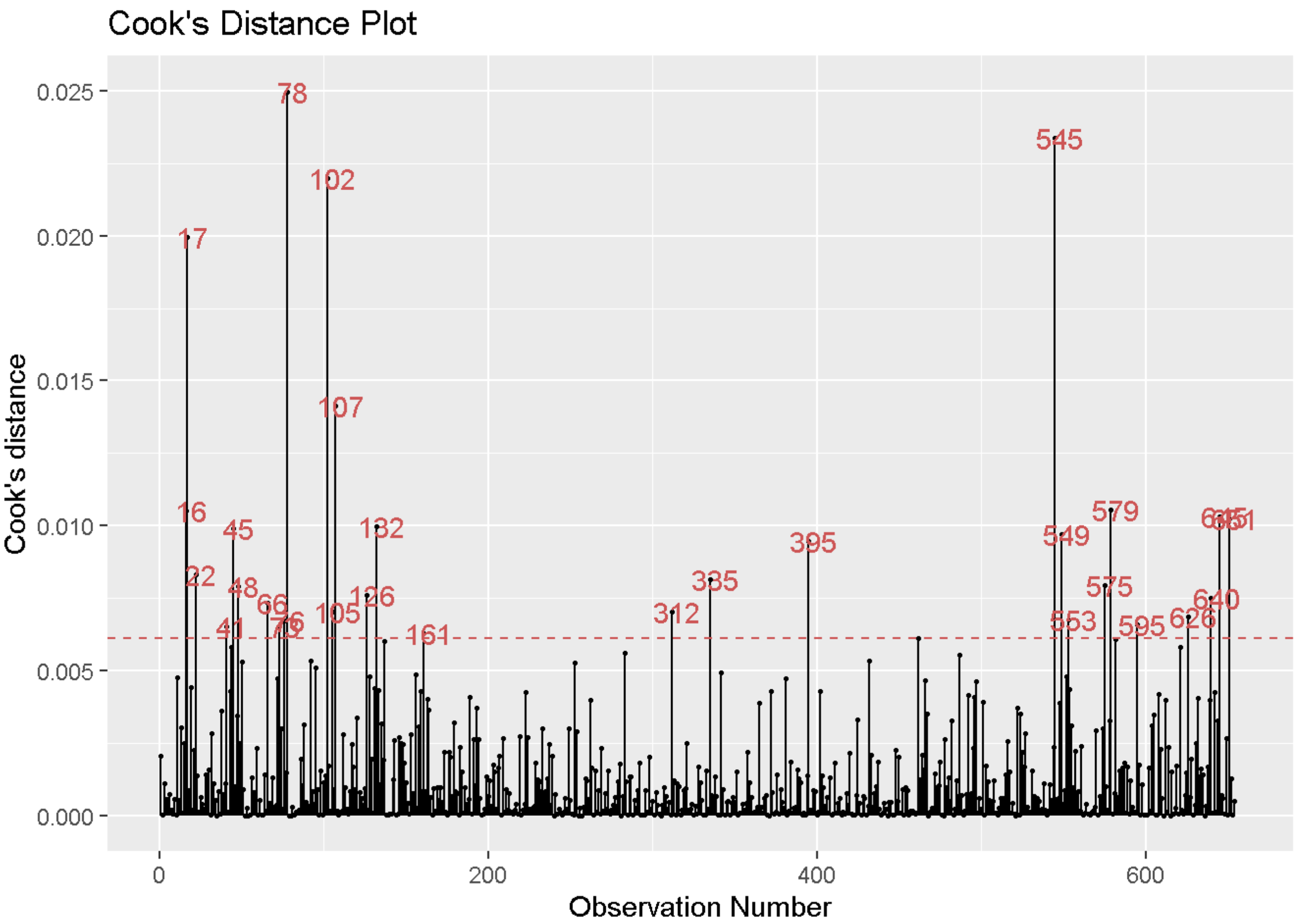


Processing math: 100%





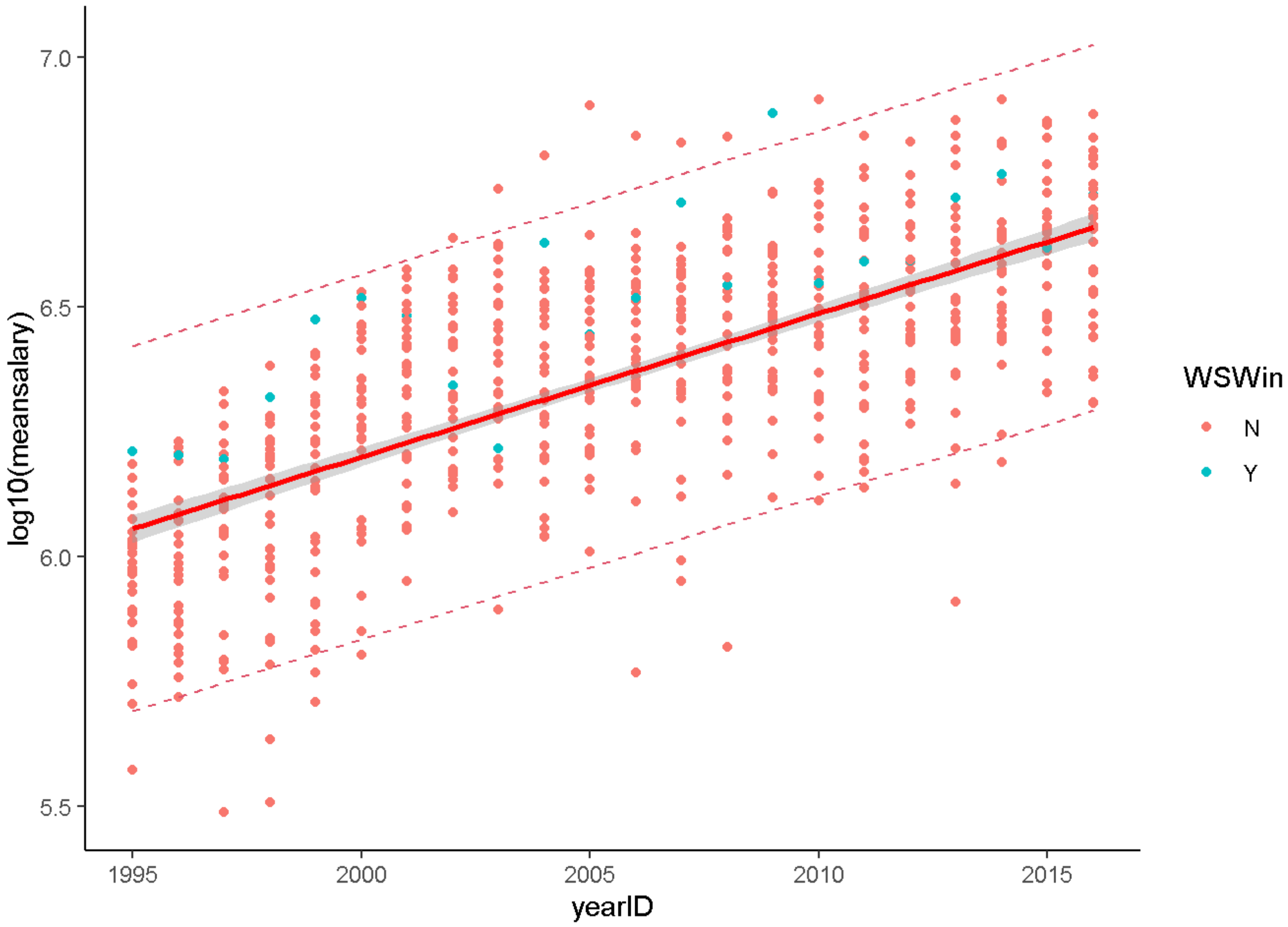
Processing math: 100%



1. Linearity: to begin with we can look at both scatter plots in question 2a which both would indicate some form of linearity at first a glance. A look at the residual plots in the diagnose function would further support this assumption, I would say that we have a strong case for linearity here.
2. Homoscedasticity: for this assumption we have to analyse the variance in residual values for our data. In the above diagnose function, looking at the “Residual vs yearID” I believe this is a reasonable residual plot to assume homoscedasticity.
3. Normality: we can look at the QQ plot in the diagnose function, and seeing as the points are almost all hugging the straight line on the plot we can say that the residuals are approximately normally distributed. This is further supported by the histogram plot which we can see clearly indicated normality.
4. Independence: as we have time series data we have to watch out for autocorrelation, i.e. are data points easily predicted or known based on the data that’s come in the previous X value/s. Looking at the residual plot for yearID, it doesn’t seem that there is any autocorrelation occurring, I would say that generally the results seem independent of each other; however it would be interesting to see if we had a larger data set in the future whether or not time based factors such as economic cycle crashes would influence this data.

(2d)

```
confint(linmod1) -> confidenceinterval1

predict(linmod1, interval = "prediction") -> predictionband1

as.tibble(predictionband1) -> predictiontibble

bind_cols(Teamdata, predictiontibble) -> TeamSalary2d

TeamSalary2d %>%
  ggplot(aes(x = yearID, y = log10(meansalary), colour = WSWin)) +
  geom_point() +
  geom_smooth(method = lm, colour = "red") +
  geom_line(aes(y = lwr), color = 2, lty = 2) +
  geom_line(aes(y = upr), color = 2, lty = 2) +
  theme_classic()
```



From this plot we can see that your chance of a world series win is definitely influenced by the amount you pay your teams, however I would also say that it seems post 2000 this is slowly becoming less of the case; perhaps due to the fact that more teams have the money to hand out (just speculating lol I have no clue about baseball).

(2e)

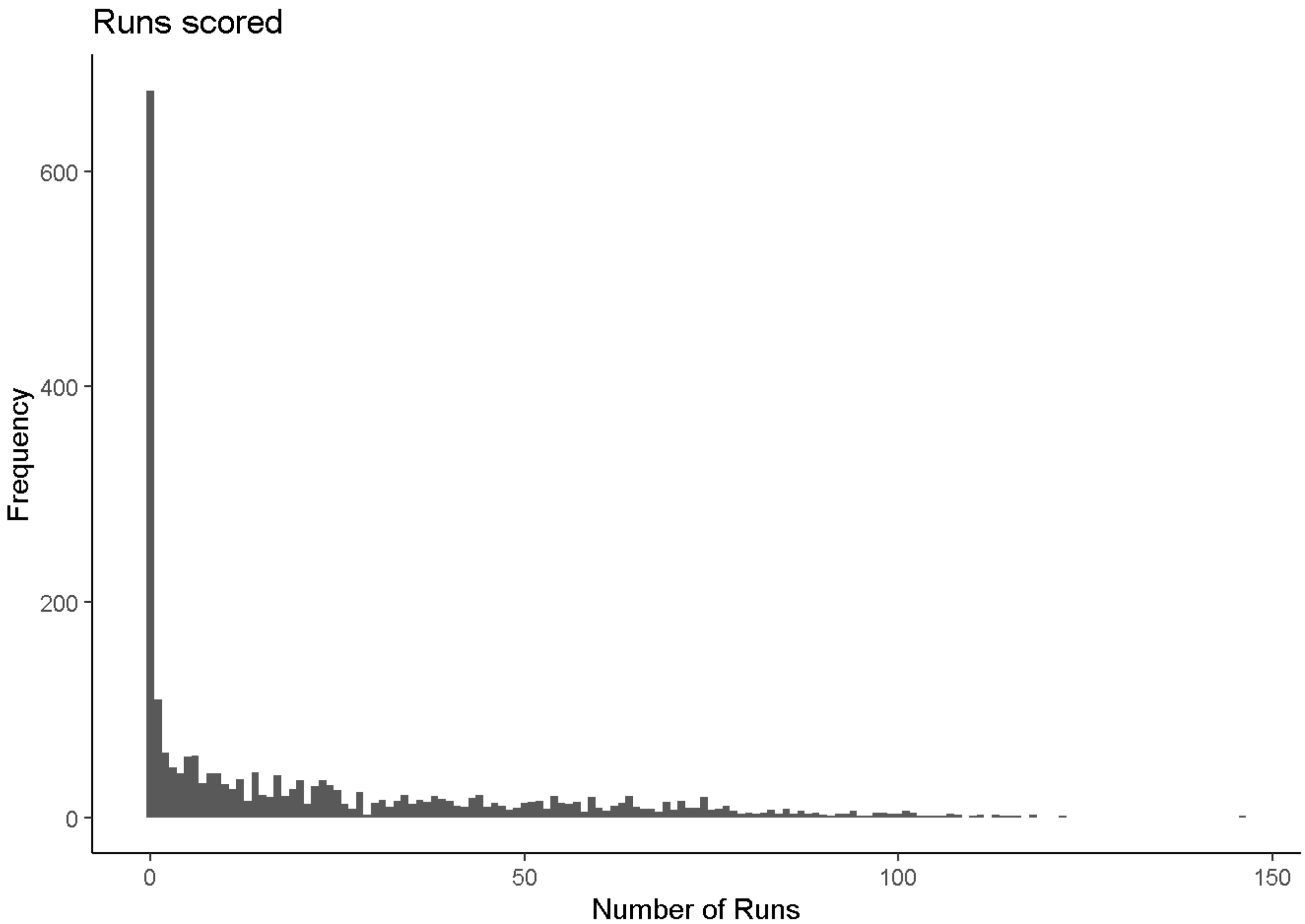
```
TeamSalary2d %>%
  filter(log10(meansalary) > upr) %>%
  select(yearID, name) %>%
  count(name)
```

```
             name n
1 New York Yankees 9
```

The teams appearing above the top prediction band are always the New York Yankees - a total of 9 times.

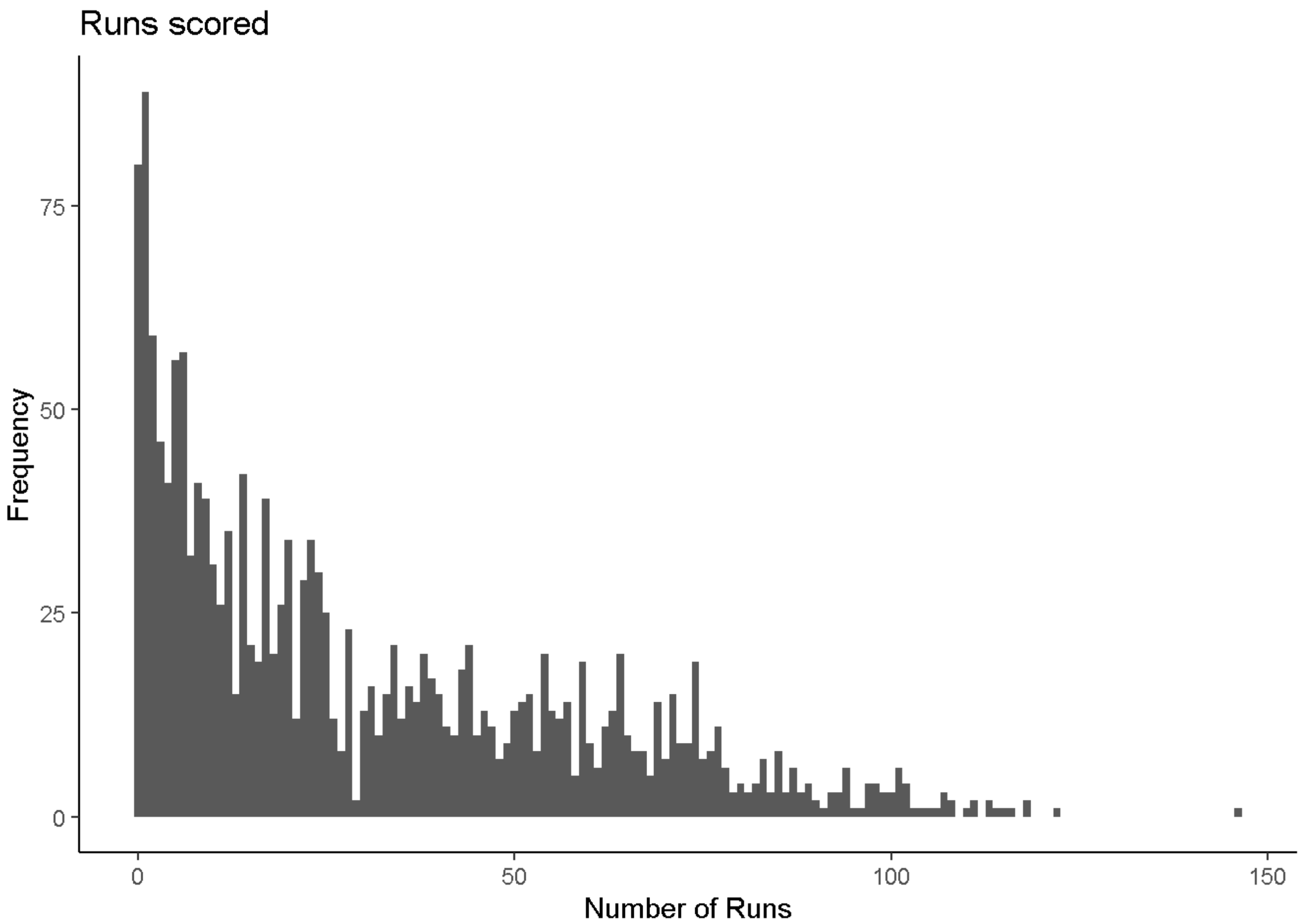
(3a)

```
Playerdata %>%
  ggplot(aes(R)) +
  geom_histogram(binwidth = 1) +
  labs(x = "Number of Runs", y = "Frequency", title = "Runs scored") +
  theme_classic() -> RunsScored

RunsScored
```




```
Playerdata %>%
  filter(H > 0) %>%
  ggplot(aes(R)) +
  geom_histogram(binwidth = 1) +
  labs(x = "Number of Runs", y = "Frequency", title = "Runs scored") +
  theme_classic() -> HitRuns

HitRuns
```

Processing math: 100%



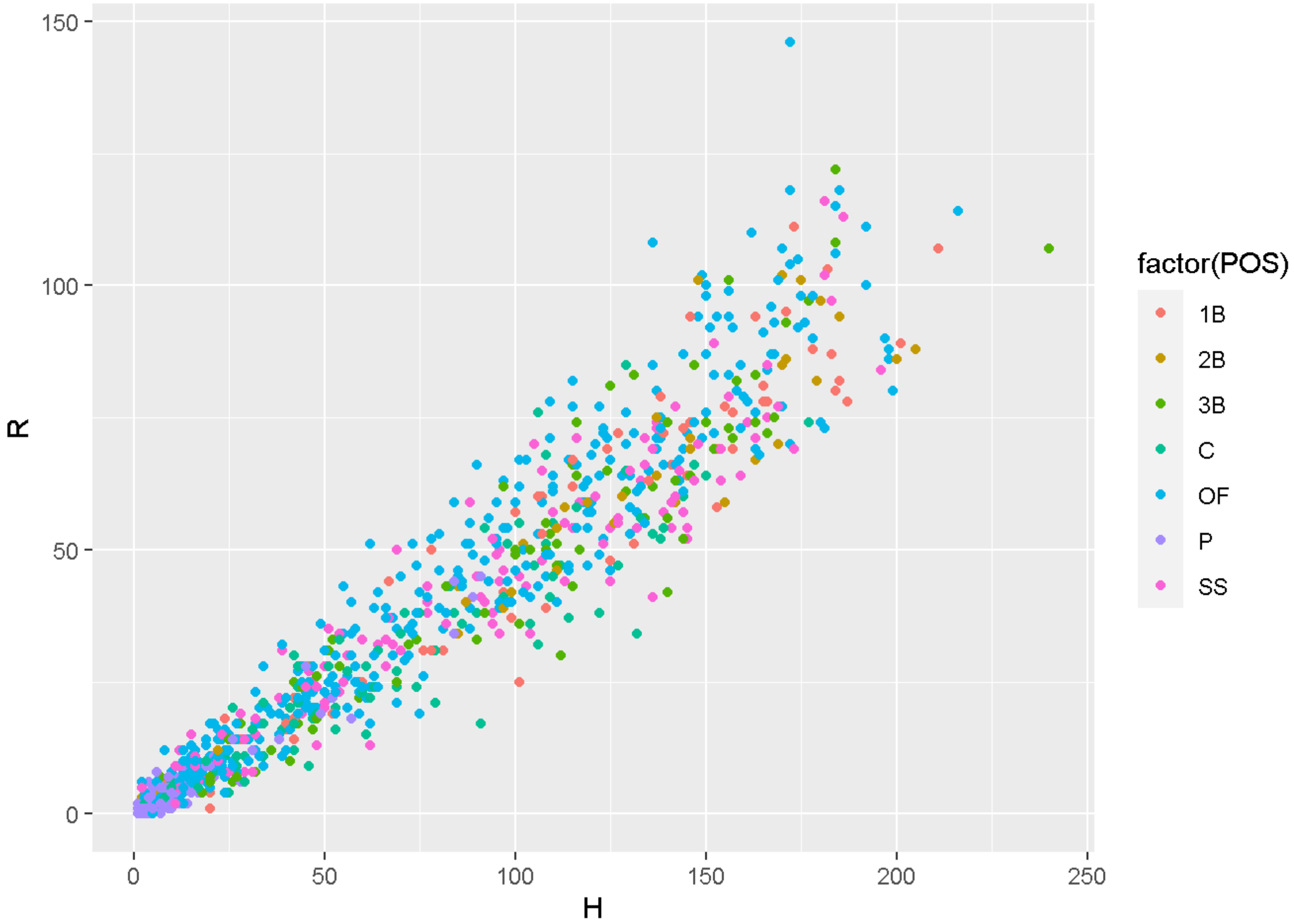
The second data set is more reasonable to use when creating our model as it is pointless to include players who haven’t had the chance to score a run in a research question looking at runs.

Including the full set of hits = 0 will skew the data visualization making it harder to read.

(3b)

```
Playerdata %>%
  filter(H > 0) %>%
  mutate("yearID" = as.factor(yearID)) -> OnBaseP

glm(formula = R ~ H + as.factor(yearID) + POS + height + age, family = "poisson", data = OnBase
P) -> glm1

glm1

# I'm going to evaluate the data we have checking whether any positions shouldn't be included i
n this model (I'm pretty sure baseball has their own version of a useless role like goalkeeper)

glm1 %>%
  ggplot(aes(x= H, y = R, colour = factor(POS))) +
  geom_point()
```

Processing math: 100%




```
# Have found that POS = P (pitchers) don't get many hits or runs (I think they only get to hit
 if you rotate through every other player on the team? And I guess because rarely do it as well
they usually suck). So I will remove Pitchers from the Hit model.

Playerdata %>%
  filter(H > 0) %>%
  mutate("yearID" = as.factor(yearID)) %>%
  filter(POS != "P") -> OnBase

glm(formula = R ~ H + as.factor(yearID) + POS + height + age, family = "poisson", data = OnBas
e) -> glm2
```

Processing math: 100%

```
Call:  glm(formula = R ~ H + as.factor(yearID) + POS + height + age,
    family = "poisson", data = OnBaseP)

Coefficients:
          (Intercept)                        H  as.factor(yearID)2015
             2.347998                 0.013332                 0.024172
                POS2B                    POS3B                     POSC
            -0.005030                 0.008146                -0.061520
                POSOF                     POSP                    POSSS
             0.069327                -1.161731                -0.014180
               height                      age
            -0.002828                 0.005467

Degrees of Freedom: 1739 Total (i.e. Null);  1729 Residual
Null Deviance:      44650
Residual Deviance: 6734     AIC: 14680
```

(3c)

```
Anova(glm2)
```

```
Analysis of Deviance Table (Type II tests)

Response: R
                  LR Chisq Df Pr(>Chisq)
H                  24778.8  1  < 2.2e-16 ***
as.factor(yearID)      1.7  1   0.197407
POS                   79.6  5  1.034e-15 ***
height                 0.0  1   0.988469
age                    6.8  1   0.008928 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

The hypothesis being tested by each p value is whether not the full model is a better method for prediction compared to a reduced model (which would be the NULL hypothesis). Each p value indicates whether or not that variable is a relevant predictor of the number of runs (R) scored.

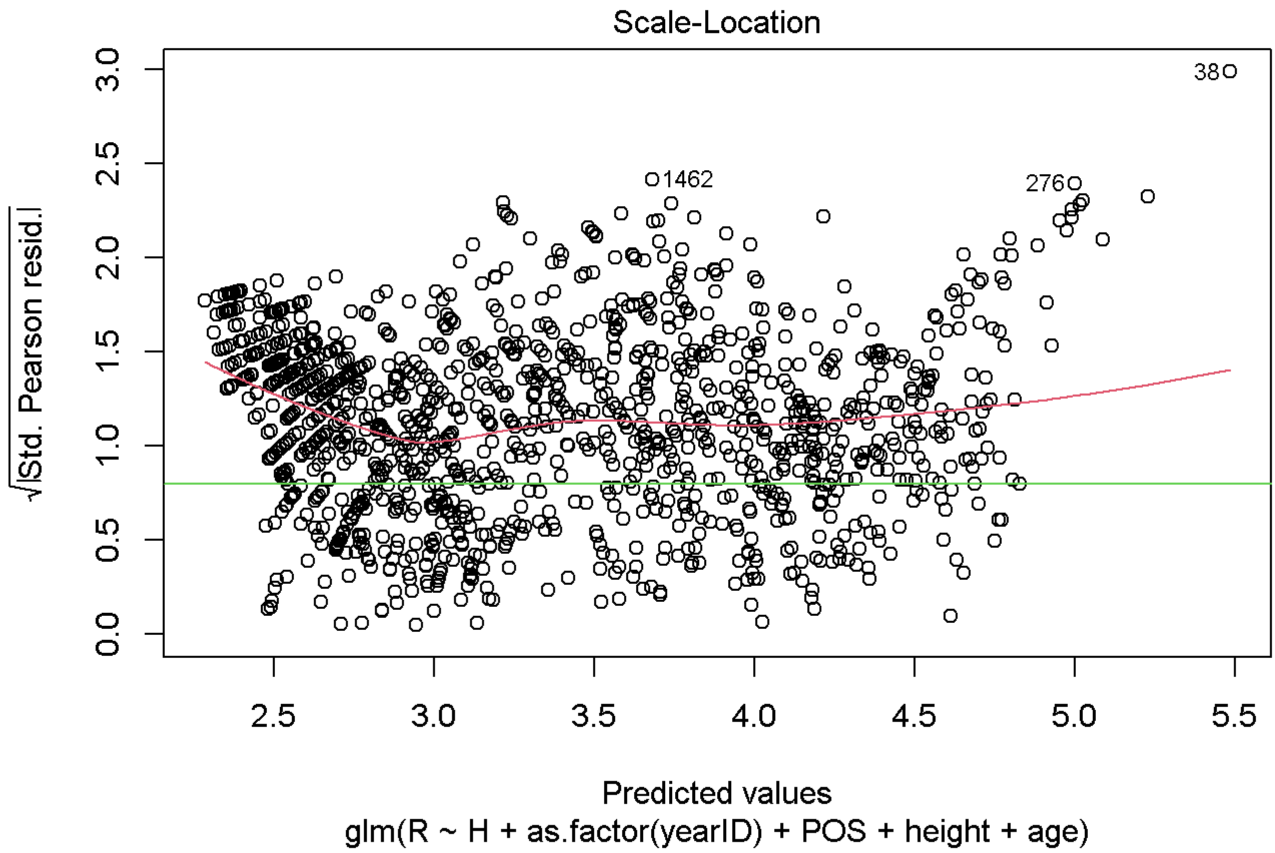
We can see that the p value for POS (1.034e-15 or 0.000000000000001034) is extremely small, almost 0. So we have strong evidence that including position within our model generates more accuracy than excluding it.

We can also see that the p value for height (0.988469) is the greatest value and is above the typical threshold of 0.05. This indicates that height is not an important predictor of runs scored.

(3d)

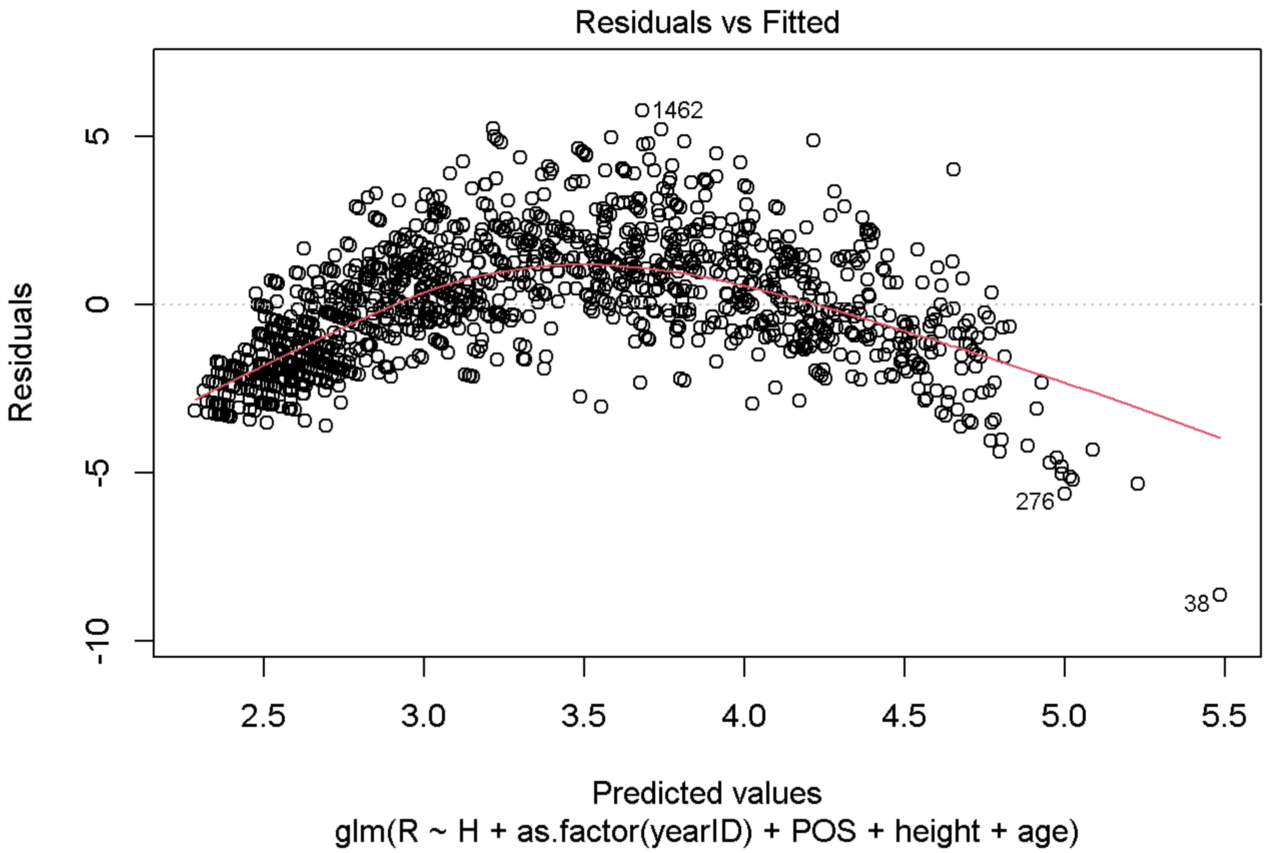
```
plot(glm2,which=3)
abline(h=0.8,col=3)
```

Processing math: 100%



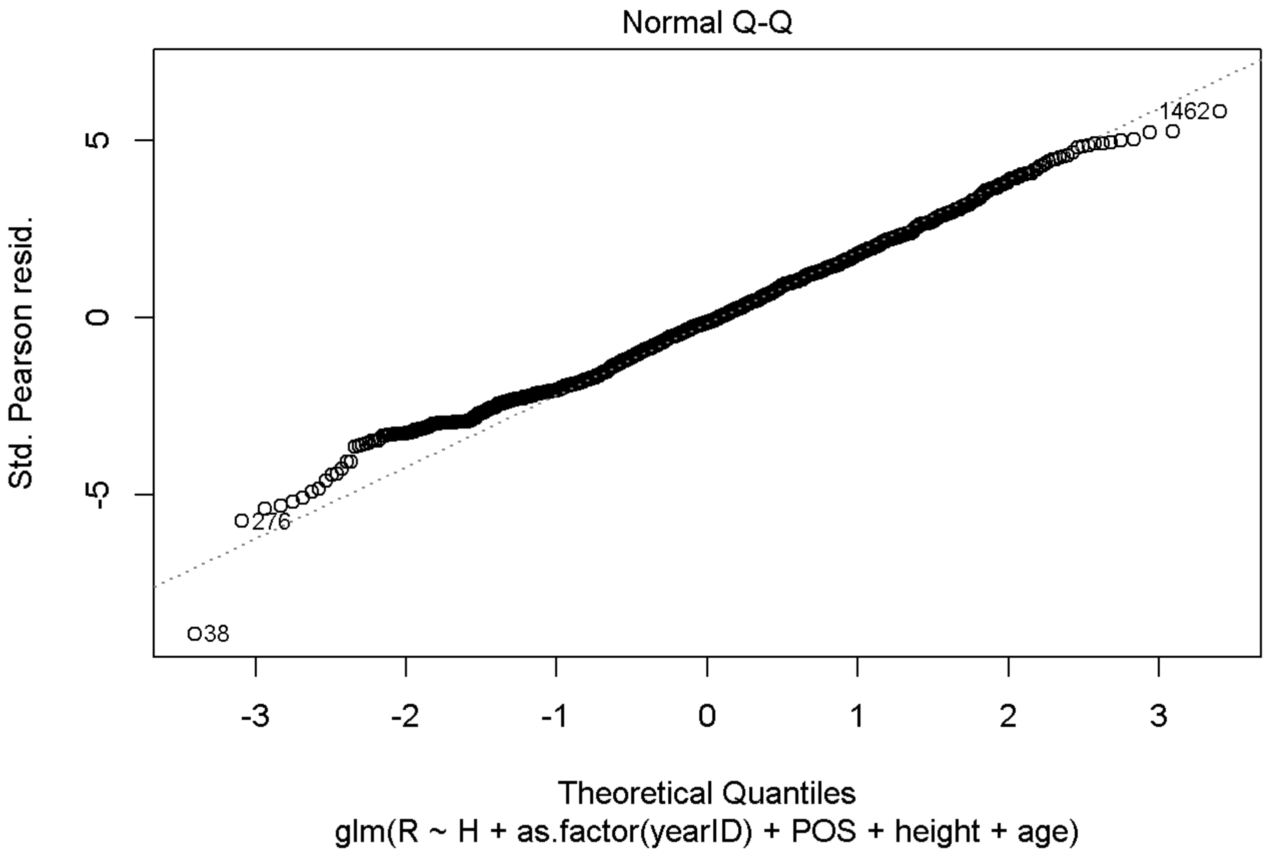

```
# Linearity

plot(glm2,which=1)
```

Processing math: 100%




```
# Poisson distribution

plot(glm2,which=2)
```

Processing math: 100%



The first assumption we can test is the dispersion assumption (testing if variance = mean), so the variance should rise when our measured values rise; winning teams with skyrocketing salaries may create an environment where the smaller teams can't keep up with the paychecks - widening the spread of data. We can see the red line, in the dispersion chart above is slightly higher than 0.8 meaning the data is overdispersed.

For our linearity assumption it seems that the data follows a parabolic curve rather than a straight line. Meaning that we will have to look for a better model and test that one.

For the distribution assumption we can look at the QQ plot and see that the data follows the line nicely so we won't have to use robust confidence intervals.

Finally there is the independence assumption, in our data set there is not natural order to how the data is collected so we cannot check this assumption. (Will data points be influenced by the data point before it)

(3e)

Processing math: 100%

```
glmer(R ~ H + yearID + POS + age + (1|teamID), data = OnBase, family = "poisson") -> mmod1

mmod1

glmer(R ~ H + yearID + POS + age + (1|teamID), data = OnBase, family = "poisson", nAGQ = 0) ->
 mmod2

mmod2

# statistical significance, there are some warnings

# glmer(R ~ H + yearID + POS + age + (1|teamID), data = OnBase, family = "poisson") -> mmod3

# confint(mmod3)
# this code takes really long to run so I commented it when knitting
```

```
Generalized linear mixed model fit by maximum likelihood (Laplace
  Approximation) [glmerMod]
 Family: poisson  ( log )
Formula: R ~ H + yearID + POS + age + (1 | teamID)
   Data: OnBase
     AIC      BIC   logLik deviance df.resid
12602.48 12655.61 -6291.24 12582.48     1490
Random effects:
 Groups Name        Std.Dev.
 teamID (Intercept) 0.1055
Number of obs: 1500, groups:  teamID, 33
Fixed Effects:
(Intercept)            H   yearID2015        POS2B        POS3B         POSC
   2.248067     0.013191     0.015829    -0.001358     0.010591    -0.064989
      POSOF        POSSS          age
   0.067566    -0.013711     0.002408
optimizer (Nelder_Mead) convergence code: 0 (OK) ; 0 optimizer warnings; 2 lme4 warnings
Generalized linear mixed model fit by maximum likelihood (Adaptive
  Gauss-Hermite Quadrature, nAGQ = 0) [glmerMod]
 Family: poisson  ( log )
Formula: R ~ H + yearID + POS + age + (1 | teamID)
   Data: OnBase
      AIC       BIC    logLik  deviance  df.resid
12602.482 12655.614 -6291.241 12582.482      1490
Random effects:
 Groups Name        Std.Dev.
 teamID (Intercept) 0.1055
Number of obs: 1500, groups:  teamID, 33
Fixed Effects:
(Intercept)            H   yearID2015        POS2B        POS3B         POSC
   2.248398     0.013191     0.015832    -0.001356     0.010591    -0.064989
      POSOF        POSSS          age
   0.067567    -0.013710     0.002408
```

Runs ~ Pois(2.248398 + 0.013191 * H + POS2B * -0.001356 + POS3B * 0.010591 + POSC * -0.064989 + POSOF * 0.067567 + POSSS * -0.013710 + age * 0.002408 + yearID2015 * 0.015832 + u)

u ~ N(0, 0.1055)

Processing math: 100%

We can see that the standard deviation of the effect of team on number of runs was 0.1055 and average runs are 2.248398.

So this means being in the top club as opposed to an average team a player will score about 0.57 more runs, just based on team as a stand alone. (e(2*0.1055)). Which is a reasonably large difference.

You can check the statistical significance of this effect by using confidence intervals.

(3f)

```
predict(mmod2, newdata=data.frame(age = 30, height = 72, teamID = "BAL", POS = "OF", yearID =
"2015", H = 20), type = "response")
```

```
       1
16.76264
```

You would expect a mean of 16.763 runs for such a player.

(4a)

```
Teamdata %>%
  select(!c(teamID, name, park, teamIDBR, teamIDlahman45, teamIDretro, lgID, Rank, franchID, di
vID, WCWin, LgWin, WSWin)) -> DivWinners

DivWinners

set.seed(123)

DivWinners$DivWin %>%
  createDataPartition(p = 0.8, list = FALSE) -> training.samples

DivWinners[training.samples, ] -> train.data
DivWinners[-training.samples, ] -> test.data
```

Processing math: 100%

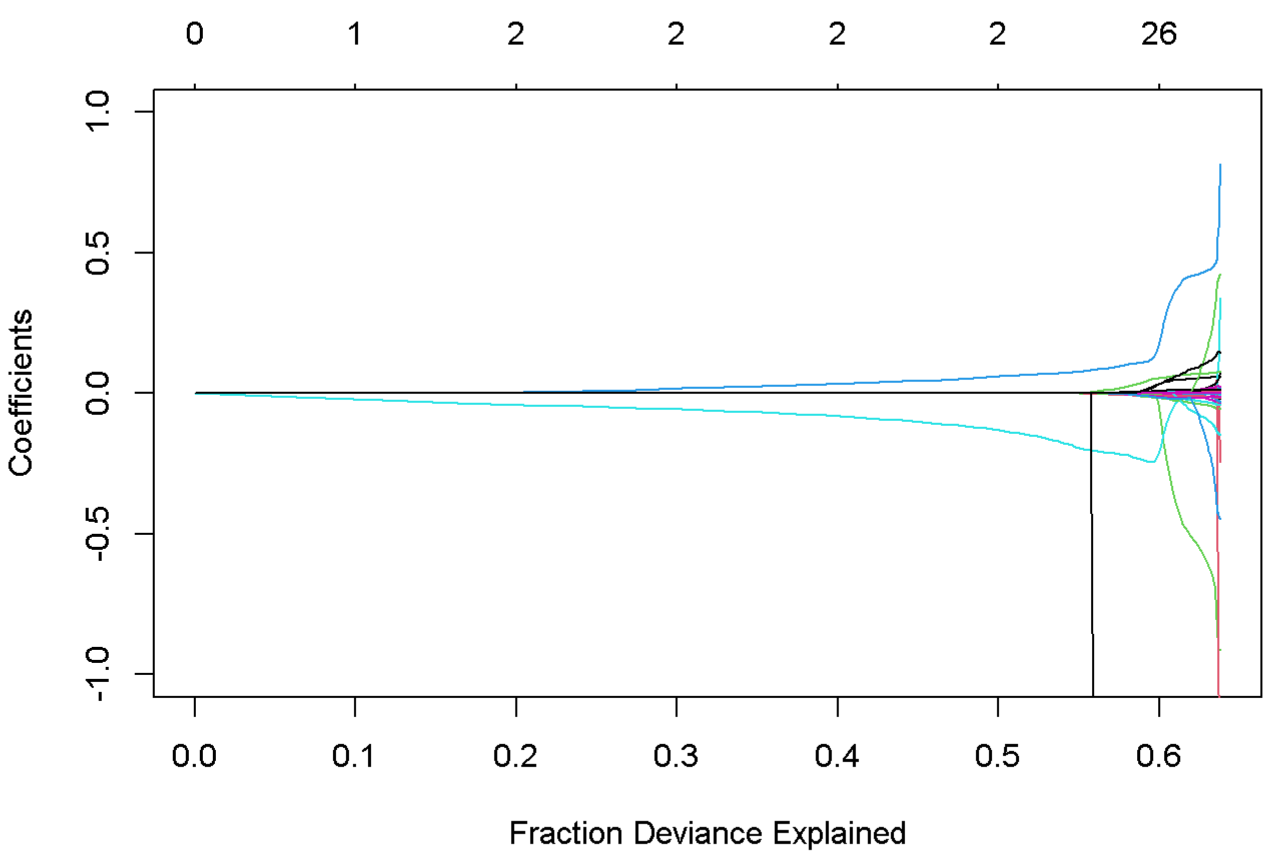
	yearID	G	Ghome	W	L	DivWin	R	AB	H	X2B	X3B	HR	BB	SO	SB	CS
1	1995	144	72	90	54	Y	645	4814	1202	210	27	168	520	933	73	43
2	1995	144	72	71	73	N	704	4837	1267	229	27	173	574	803	92	45
3	1995	144	72	86	58	Y	791	4997	1399	286	31	175	560	923	99	44
4	1995	145	72	78	67	N	801	5019	1390	252	25	186	564	889	58	39
5	1995	145	72	68	76	N	755	5060	1417	252	37	146	576	767	110	39
6	1995	144	72	73	71	N	693	4963	1315	267	39	158	440	953	105	37
7	1995	144	72	85	59	Y	747	4903	1326	277	35	161	519	946	190	68
8	1995	144	72	100	44	Y	840	5028	1461	279	23	207	542	766	132	53
9	1995	144	72	77	67	N	785	4994	1406	259	43	200	484	943	125	59
10	1995	144	72	60	84	N	654	4865	1204	228	29	159	551	987	73	36
11	1995	143	71	67	76	N	673	4886	1278	214	29	144	517	916	131	53
12	1995	144	72	76	68	N	747	5097	1403	260	22	109	566	992	176	60
13	1995	144	72	70	74	N	629	4903	1275	240	35	119	475	849	120	53
14	1995	144	72	78	66	Y	634	4942	1303	191	31	140	468	1023	127	45
15	1995	144	72	56	88	N	703	5005	1398	270	34	120	471	916	105	57
16	1995	144	72	65	79	N	740	5000	1329	249	42	128	502	800	105	40
17	1995	144	72	66	78	N	621	4905	1268	265	24	118	400	901	120	49
18	1995	145	73	79	65	N	749	4947	1365	280	34	122	625	851	50	30
19	1995	144	72	69	75	N	657	4958	1323	218	34	125	446	994	58	39
20	1995	144	72	67	77	N	730	4916	1296	228	18	169	565	911	112	46
21	1995	144	72	69	75	N	615	4950	1296	263	30	94	497	884	72	25
22	1995	144	72	58	86	N	629	4937	1281	245	27	125	456	972	84	55
23	1995	144	72	70	74	N	668	4950	1345	231	20	116	447	872	124	46
24	1995	145	73	79	66	Y	796	4996	1377	276	20	182	549	871	110	41
25	1995	144	72	67	77	N	652	4971	1256	229	33	152	472	1060	138	46
26	1995	143	72	62	81	N	563	4779	1182	238	24	107	436	920	79	46
27	1995	144	72	74	70	N	691	4913	1304	247	24	138	526	877	90	47
28	1995	144	72	56	88	N	642	5036	1309	275	27	140	492	906	75	16
29	1996	162	81	96	66	Y	773	5614	1514	264	28	197	530	1032	83	43
30	1996	163	82	88	74	N	949	5689	1557	299	29	257	645	915	76	40
31	1996	162	81	85	77	N	928	5756	1631	308	31	209	642	1020	91	44
32	1996	161	81	70	91	N	762	5686	1571	256	24	192	527	974	53	39
33	1996	162	81	85	77	N	898	5644	1586	284	33	195	701	927	105	41
34	1996	162	81	76	86	N	772	5531	1388	267	19	175	523	1090	108	50
35	1996	162	81	81	81	N	778	5455	1398	259	36	191	604	1134	171	63
36	1996	161	80	99	62	Y	952	5681	1665	335	23	218	671	844	160	50
37	1996	162	81	83	79	N	961	5590	1607	297	37	221	527	1108	201	66
38	1996	162	81	53	109	N	783	5530	1413	257	21	204	546	1268	87	50
39	1996	162	81	80	82	N	688	5498	1413	240	30	150	553	1122	99	46
40	1996	162	81	82	80	N	753	5508	1445	297	29	129	554	1057	180	63
41	1996	161	80	75	86	N	746	5542	1477	286	38	123	529	943	195	85
42	1996	162	81	90	72	N	703	5538	1396	215	33	150	516	1190	124	40
43	1996	162	82	78	84	N	877	5673	1633	332	47	118	576	958	143	53
44	1996	162	81	80	82	N	894	5662	1578	304	40	178	624	986	101	48
45	1996	162	81	88	74	N	741	5505	1441	297	27	148	492	1077	108	34
46	1996	162	80	92	70	Y	871	5628	1621	293	28	162	632	909	96	46
47	1996	162	81	71	91	N	746	5618	1515	267	47	147	445	1069	97	48
48	1996	162	81	78	84	N	861	5630	1492	283	21	243	640	1114	58	35
49	1996	162	81	67	95	N	650	5499	1405	249	39	132	536	1092	117	41
50	1996	162	80	73	89	N	776	5665	1509	319	33	138	510	989	126	49
51	1996	162	81	91	71	Y	771	5655	1499	285	24	147	601	1014	109	55
52	1996	161	81	85	76	N	993	5668	1625	343	19	245	670	1052	90	39
53	1996	162	82	68	94	N	752	5533	1400	245	21	153	615	1189	113	53
54	1996	162	81	88	74	Y	759	5502	1468	281	31	142	495	1089	149	58

Processing math: 100%

```
649     3365256 103 102  172253778  6890151.1          25
650     3444490 100  99  143053500  4614629.0          31
651     1286163  93  94   57097310  2039189.6          28
652     2710402 106 105  176038723  6070300.8          29
653     3392099 111 110  138701700  4782817.2          29
654     2481938 100  98  141652646  5448178.7          26
```

(4b)

```
as.vector(train.data$DivWin) -> divwin1
model.matrix(~ . -1, train.data[,-c(6)]) -> divpredict1

glmnet(divpredict1, divwin1, family = "binomial") -> divwinfit

plot(divwinfit, xvar = "dev",ylim = c(0,0))
```



(4c)

Processing math: 100%

```
divwinfit

# First over 0.5: 21  2 50.02 0.038030
# First over 0.6: 53 26 60.13 0.001937

coef(divwinfit, s = 0.038030) -> divwin4c50
divwin4c50@Dimnames[[1]][1+divwin4c50@i]

coef(divwinfit, s = 0.001937) -> divwin4c60
divwin4c60@Dimnames[[1]][1+divwin4c60@i]
```

Processing math: 100%

```

Call:  glmnet(x = divpredict1, y = divwin1, family = "binomial")

   Df  %Dev   Lambda
1   0  0.00 0.244500
2   1  6.30 0.222800
3   1 11.67 0.203000
4   1 16.31 0.184900
5   2 20.45 0.168500
6   2 24.13 0.153500
7   2 27.40 0.139900
8   2 30.31 0.127500
9   2 32.92 0.116100
10  2 35.27 0.105800
11  2 37.38 0.096430
12  2 39.29 0.087860
13  2 41.02 0.080060
14  2 42.58 0.072940
15  2 44.00 0.066460
16  2 45.27 0.060560
17  2 46.43 0.055180
18  2 47.47 0.050280
19  2 48.41 0.045810
20  2 49.26 0.041740
21  2 50.02 0.038030
22  2 50.70 0.034650
23  3 51.31 0.031580
24  3 51.91 0.028770
25  3 52.45 0.026220
26  3 52.92 0.023890
27  3 53.34 0.021760
28  3 53.71 0.019830
29  3 54.04 0.018070
30  3 54.33 0.016460
31  3 54.58 0.015000
32  4 54.80 0.013670
33  6 55.04 0.012450
34  7 55.29 0.011350
35  8 55.74 0.010340
36  9 56.15 0.009421
37  9 56.51 0.008584
38 12 56.84 0.007822
39 14 57.18 0.007127
40 14 57.52 0.006494
41 15 57.81 0.005917
42 15 58.10 0.005391
43 16 58.36 0.004912
44 18 58.61 0.004476
45 19 58.84 0.004078
46 19 59.04 0.003716
47 19 59.21 0.003386
48 20 59.36 0.003085
49 21 59.49 0.002811
50 22 59.63 0.002561
51 23 59.76 0.002334
```

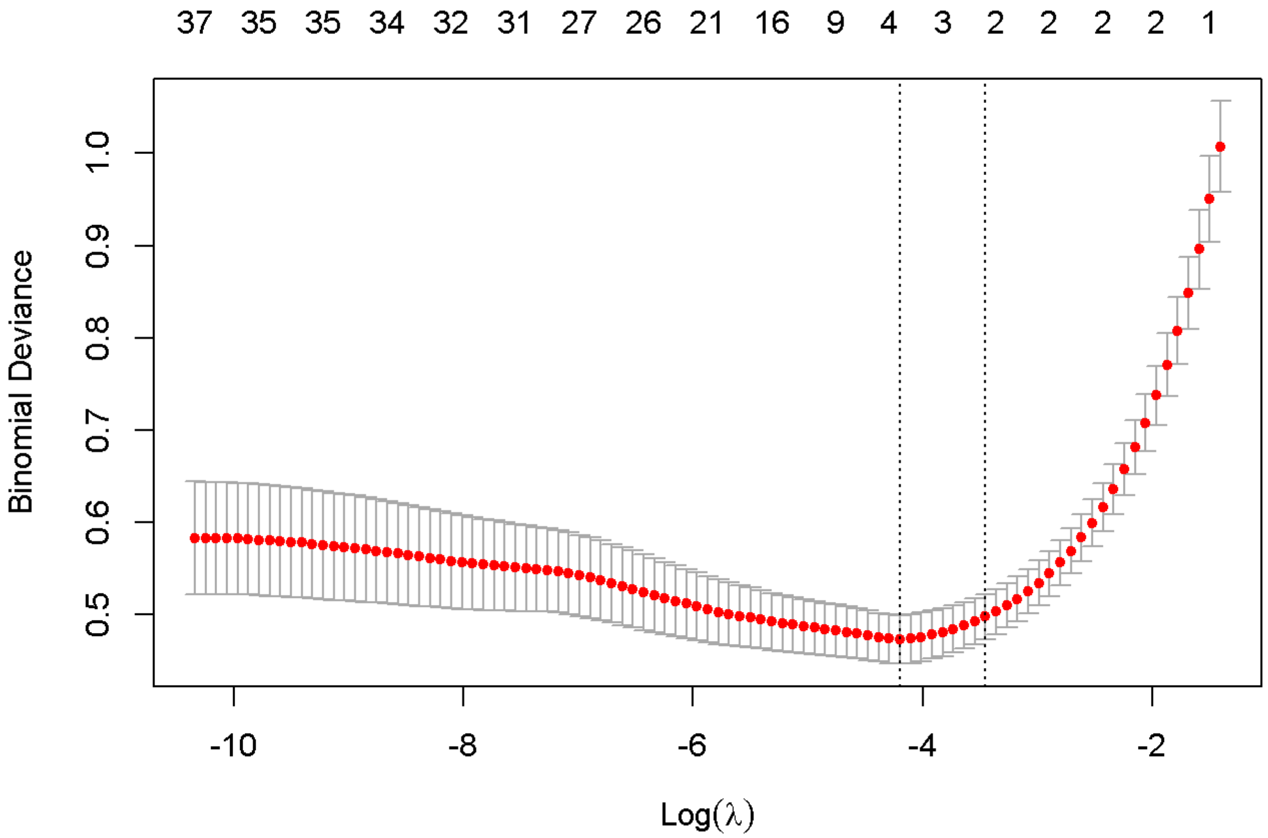
```
52 25 59.93 0.002126
53 26 60.13 0.001937
54 27 60.29 0.001765
55 26 60.44 0.001609
56 27 60.61 0.001466
57 27 60.80 0.001335
58 28 61.04 0.001217
59 28 61.28 0.001109
60 28 61.48 0.001010
61 27 61.62 0.000920
62 29 61.79 0.000839
63 29 61.98 0.000764
64 30 62.21 0.000696
65 31 62.42 0.000634
66 31 62.60 0.000578
67 31 62.76 0.000527
68 31 62.89 0.000480
69 31 63.00 0.000437
70 32 63.10 0.000398
71 32 63.18 0.000363
72 32 63.25 0.000331
73 32 63.30 0.000301
74 32 63.35 0.000275
75 33 63.40 0.000250
76 33 63.43 0.000228
77 34 63.46 0.000208
78 34 63.49 0.000189
79 34 63.51 0.000172
80 34 63.53 0.000157
81 34 63.55 0.000143
82 34 63.56 0.000130
83 34 63.57 0.000119
84 35 63.58 0.000108
85 35 63.59 0.000099
86 35 63.60 0.000090
87 35 63.61 0.000082
88 36 63.62 0.000075
89 36 63.65 0.000068
90 35 63.66 0.000062
91 35 63.67 0.000056
92 35 63.68 0.000051
93 36 63.69 0.000047
94 37 63.73 0.000043
95 37 63.78 0.000039
96 37 63.81 0.000035
97 37 63.81 0.000032
[1] "(Intercept)" "W"           "L"
 [1] "(Intercept)" "yearID"      "Ghome"       "W"           "L"
 [6] "AB"          "H"           "X2B"         "X3B"         "HR"
[11] "BB"          "SO"          "SB"          "CS"          "HBP"
[16] "SF"          "RA"          "CG"          "SV"          "HA"
[21] "HRA"         "BBA"         "SOA"         "DP"          "FP"
[26] "attendance"  "PPF"         "rostersize"
```

Processing math: 100%

We can see from the plot that 50% of the deviance can be explained by 2 parameters(Wins(W) and Losses(L)), and 60% can be explained by 26 parameters (“yearID”, “Ghome”, “W”, “L”, “AB”, “H”, “X2B”, “X3B”, “HR”, “BB”, “SO”, “SB”, “CS”, “HBP”, “SF”, “RA”, “CG”, “SV”, “HA”, “HRA”, “BBA”, “SOA”, “DP”, “FP”, “attendance”, “PPF”, “roster size”.)

(4d)

```
set.seed(312)
cv.glmnet(divpredict1, divwin1, family = "binomial") -> crossvalidation
plot(crossvalidation)
```




```
coef(divwinfit, s = crossvalidation$lambda.1se) -> divwincoef

divwincoef

setdiff(divwincoef@Dimnames[[1]][1+divwincoef@i], divwin4c50@Dimnames[[1]][1+divwin4c50@i])
```

Processing math: 100%

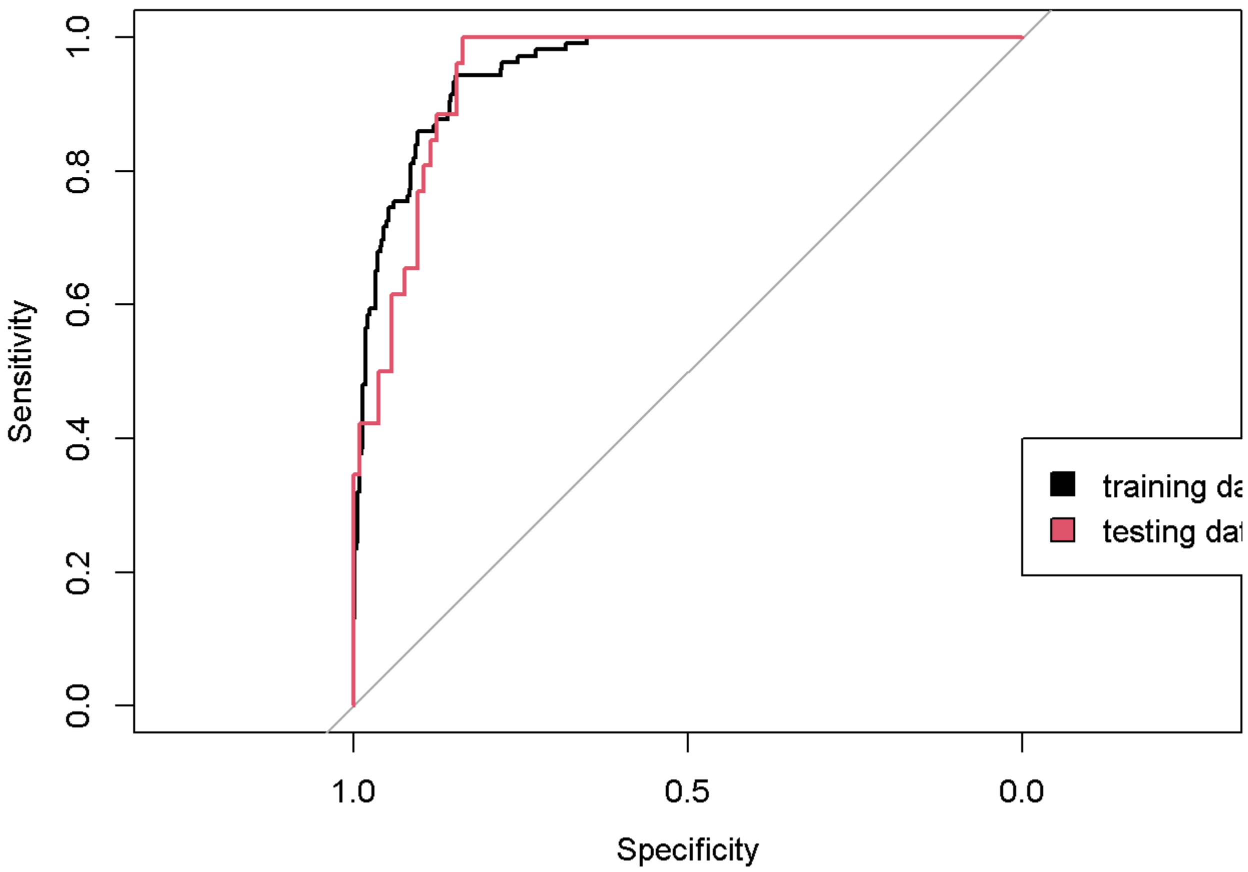
```
38 x 1 sparse Matrix of class "dgCMatrix"
                        s1
(Intercept)  3.674316e+00
yearID       .
G            .
Ghome        .
W            6.536722e-02
L           -1.417659e-01
R            .
AB           .
H            .
X2B          .
X3B          .
HR           .
BB           .
SO           .
SB           .
CS           .
HBP          .
SF           .
RA           .
ER           .
ERA          .
CG           .
SHO          .
SV           .
IPouts       .
HA           .
HRA          .
BBA          .
SOA          .
E            .
DP           .
FP           .
attendance   1.444007e-10
BPF          .
PPF          .
Rostercost   .
meansalary   .
rostersize   .
[1] "attendance"
```

Looking at the plot of cross validation we can see that around 3~5 variables seems to be where the binomial deviance is at its lowest. So going off this I have cross validated against our previous previous analysis where s = 0.038030 (lamda value showing 2 variables) as this is closer to our amount of values suggested here. Looking at our coefficient results we can see that “attendance” is suggested to be included within the model. So for my conservative model I will include: W, L, attendance.

(4e)

Processing math: 100%

```
DivWinners
train.data
test.data

glm(as.factor(DivWin) ~ W + L + attendance, data = train.data, family = "binomial") -> train.mo
del4

train.model4 %>%
  predict(type = "response") -> predtrain

train.model4 %>%
  predict(newdata = test.data, type = "response") -> predtest

roc(response = train.data$DivWin, predictor = predtrain, plot = TRUE, auc = TRUE) -> roctrain

roc(response = test.data$DivWin, predictor = predtest, plot = TRUE, auc = TRUE, add = TRUE, col
= 2)
legend(0,0.4,legend=c("training data","testing data"),fill=1:2)
```



Processing math: 100%

```
583    2148808  94  95   85556990  3168777.4         27
585    2423852 100 101  180944967  5654530.2         32
595    2080145 107 106   61834000  2132206.9         29
604    2726048  97  98  172284750  6891390.0         25
605    2153585  97  99   72256200  2580578.6         28
606    2708549 104 103  112107025  4152112.0         27
615    1831080  98  98  111693000  4295884.6         26
616    2498596  99  97   88892499  3065258.6         29
618    2193581  92  94  122208700  4888348.0         25
628    2955434 108 106  188545761  6501578.0         29
637    3016142  95  95  137251333  5278897.4         26
644    1521506  90  91   86806234  2893541.1         30
647    2351422  99  99  101424814  3756474.6         27
652    2710402 106 105  176038723  6070300.8         29

Call:
roc.default(response = test.data$DivWin, predictor = predtest,     auc = TRUE, plot = TRUE, add
= TRUE, col = 2)

Data: predtest in 104 controls (test.data$DivWin N) < 26 cases (test.data$DivWin Y).
Area under the curve: 0.9442
```

We can see that these ROC curves are very close, almost on top of each other, so this indicates that the model did not overfit the data.

(4f)

```
coords(roctrain, "b", best.method = "youden", transpose = TRUE) -> youdenroctrain

youdenroctrain

ifelse(predict(train.model4, newdata = train.data, type = "response") >= 0.1836071, "Y", "N") -
> confusiontrain

table(confusiontrain, train.data$DivWin)

ifelse(predict(train.model4, newdata = test.data, type = "response") >= 0.1836071, "Y", "N") ->
test.data$confusiontest

table(test.data$confusiontest, test.data$DivWin)
```

```
  threshold specificity sensitivity
  0.1836071   0.8468900   0.9433962

confusiontrain   N   Y
             N 354   6
             Y  64 100

    N  Y
  N 87  1
  Y 17 25
```

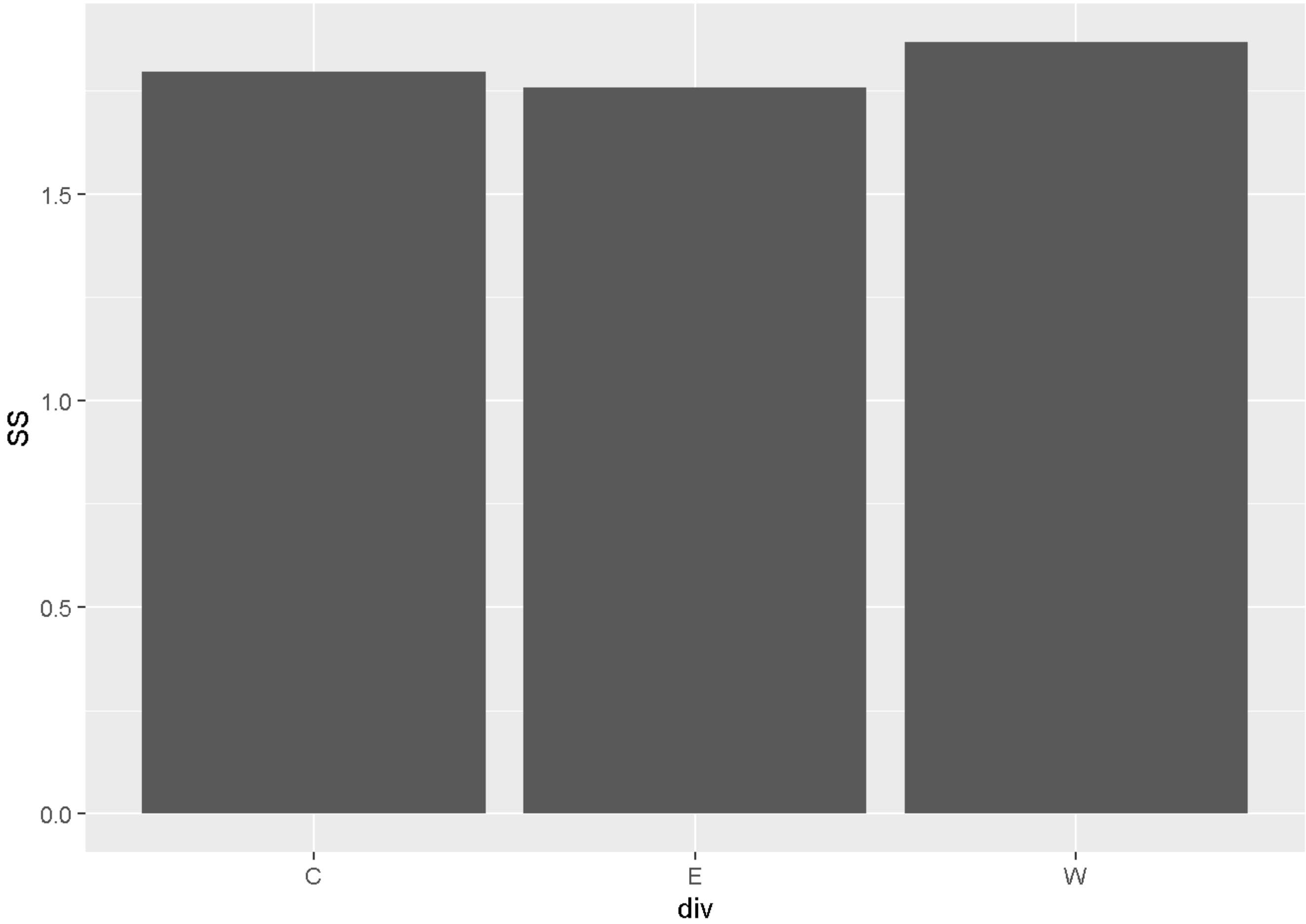
Processing math: 100%

Looking at the confusion matrix for testing data we can see that 25/26 = 96.15% DivWinners were identified correctly (Y). We can also see that 87/104 = 83.65% of Non-DivWinners were identified correctly. So this is a great model in terms of both false positives and false negatives.

(4g)

```
merge(x = test.data, y = Teamdata[ , c("yearID","W","L","DivWin","attendance","divID")], by = c
("yearID","W","L","DivWin","attendance"), all.x=TRUE) -> testdivID

testdivID %>%
  filter(divID == "C") -> testdivIDC

testdivID %>%
  filter(divID == "W") -> testdivIDW

testdivID %>%
  filter(divID == "E") -> testdivIDE


table(testdivIDC$confusiontest, testdivIDC$DivWin) -> tableC

sensitivity(tableC) + specificity(tableC) -> divC

table(testdivIDW$confusiontest, testdivIDW$DivWin) -> tableW

sensitivity(tableW) + specificity(tableW) -> divW

table(testdivIDE$confusiontest, testdivIDE$DivWin) -> tableE

sensitivity(tableE) + specificity(tableE) -> divE

data.frame(div = c("C", "W", "E"),
           SS = c(1.794715  , 1.866667, 1.757576)) -> divdf

divdf %>%
  ggplot(aes(x = div, y = SS)) +
  geom_col()
```

Processing math: 100%



Processing math: 100%