Stat 427/627 Statistical Machine Learning

In-class Lab 1: Overview of Statistical Learning

Contents

Create a RMarkdown or Quarto file and knit to Word (or HTML, pdf). The knitted file will include - Formatted text. - R code. - R output, including tables and graphs.

For the formatted text, you only need to copy-and-paste a few section to get a sense.

For the R code, you can write your own, or copy-and-paste the examples. Run them one section at a time and see if you get the same output.

Take notes in the R Markdown or Quarto file if needed.

Example 1. Kentucky Derby Data

The Kentucky Derby is a 1.25-mile horse race held annually since 1875 at the Churchill Downs racetrack in Louisville, Kentucky on the first Saturday of May. It is the first leg of the American Triple Crown. The data set KentuckyDerby.csv consists of the winners from each of the years 1875-2019. Included in the file are the Year of the race, the Winner, the Surface type (note: it has always been `dirt`), the Condition of the racing surface (1 represents a `fast` surface and 0 is not, which includes wet, muddy, and other conditions that are considered `slow`), the number of horses racing in Starters, the PolePosition of the winner, the winning Time, and the winning Speed (in miles per hour).

For this example interest is in characterizing how winning speeds (Speed) have changed since 1875 (i.e. over Year) and whether this performance depends on the track Condition. We will start with an exploratory analysis of some of the variables.

0. Load the ggplot2 library for the plots (and any other library you want to use, such as mosaic).

R comes with a lot of built in functions but R users also write their own and publish them in packages. These are very useful! `ggplot2` is a great for making very attractive plots. Base R has its own plot functions. I will illustrate both in this document.

```
# install.packages("ggplot2") # Uncomment this line if you have never used ggplot2 before!
library(ggplot2)

# install.packages("tidyverse") # uncomment this line if you have never used tidyverse before!
library(tidyverse)
```

1. Load (import) the Kentucky Derby data into R.

Note that this presumes you have set the working directory to point to your data set of interest. My shortcut is to always keep my R script in the same directory as the data set of interest and to only start R/RStudio by opening the R script.

```
derby <- read.csv("KentuckyDerby.csv", header=T)
```

2. What are the the variables in the data set? Which are quantitative and which are qualitative?

There are several ways to get at this. The `head()` function will print the first part of your data. The `names()` function will list the variable (i.e. column) names of your data. `names()` has much broader use that is helpful in other settings in R! Use it anytime you want to know what is in an object. If you use the `tidyverse` package, try `glimpse()` too.

```
head(derby)
```

```
##   Year      Winner Surface Condition Starters PolePosition   Time    Speed
## 1 1875   Aristides    dirt         1       15            1 157.75 34.23138
## 2 1876     Vagrant    dirt         1       11            4 158.35 34.10167
## 3 1877 Baden-Baden    dirt         1       11            5 158.00 34.17722
## 4 1878     DayStar    dirt         0        9            1 157.25 34.34022
## 5 1879  LordMurphy    dirt         1        9            4 157.00 34.39490
## 6 1880       Fonso    dirt         0        5            1 157.50 34.28571
```

```
names(derby)
```

```
## [1] "Year"         "Winner"       "Surface"      "Condition"    "Starters"
## [6] "PolePosition" "Time"         "Speed"
```

```
summary(derby)
```

```
##       Year         Winner            Surface            Condition
##  Min.   :1875   Length:145         Length:145         Min.   :0.0000
##  1st Qu.:1911   Class :character   Class :character   1st Qu.:0.0000
##  Median :1947   Mode  :character   Mode  :character   Median :1.0000
##  Mean   :1947                                         Mean   :0.6828
##  3rd Qu.:1983                                         3rd Qu.:1.0000
##  Max.   :2019                                         Max.   :1.0000
##     Starters      PolePosition         Time           Speed
##  Min.   : 3.00   Min.   : 1.000   Min.   :119.4   Min.   :31.35
##  1st Qu.: 9.00   1st Qu.: 3.000   1st Qu.:122.2   1st Qu.:35.02
##  Median :14.00   Median : 5.000   Median :124.0   Median :36.29
##  Mean   :13.27   Mean   : 6.683   Mean   :129.4   Mean   :35.90
##  3rd Qu.:18.00   3rd Qu.:10.000   3rd Qu.:128.5   3rd Qu.:36.82
##  Max.   :23.00   Max.   :20.000   Max.   :172.2   Max.   :37.69
```

```
derby$Condition <- factor(derby$Condition)
derby$PolePosition <- factor(derby$PolePosition)
summary(derby)
```

```
##       Year         Winner            Surface          Condition    Starters
##  Min.   :1875   Length:145         Length:145         0:46      Min.   : 3.00
##  1st Qu.:1911   Class :character   Class :character   1:99      1st Qu.: 9.00
##  Median :1947   Mode  :character   Mode  :character             Median :14.00
##  Mean   :1947                                                  Mean   :13.27
##  3rd Qu.:1983                                                  3rd Qu.:18.00
```

```
##  Max.   :2019                                                  Max.   :23.00
##
##   PolePosition      Time           Speed
##  1      :18    Min.   :119.4   Min.   :31.35
##  4      :16    1st Qu.:122.2   1st Qu.:35.02
##  5      :16    Median :124.0   Median :36.29
##  2      :14    Mean   :129.4   Mean   :35.90
##  8      :13    3rd Qu.:128.5   3rd Qu.:36.82
##  3      :11    Max.   :172.2   Max.   :37.69
##  (Other):57
```

```
glimpse(derby)  # comment out this line if tidyverse is not loaded yet.
```

```
## Rows: 145
## Columns: 8
## $ Year         <int> 1875, 1876, 1877, 1878, 1879, 1880, 1881, 1882, 1883, 188~
## $ Winner       <chr> "Aristides", "Vagrant", "Baden-Baden", "DayStar", "LordMu~
## $ Surface      <chr> "dirt", "dirt", "dirt", "dirt", "dirt", "dirt", "dirt", "~
## $ Condition    <fct> 1, 1, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, ~
## $ Starters     <int> 15, 11, 11, 9, 9, 5, 6, 14, 7, 9, 10, 10, 7, 7, 8, 6, 4, ~
## $ PolePosition <fct> 1, 4, 5, 1, 4, 1, 2, 8, 2, 8, 8, 4, 3, 6, 3, 5, 4, 2, 2, ~
## $ Time         <dbl> 157.75, 158.35, 158.00, 157.25, 157.00, 157.50, 160.00, 1~
## $ Speed        <dbl> 34.23138, 34.10167, 34.17722, 34.34022, 34.39490, 34.2857~
```

3. How many rows are in the data table? Is this what you expect?

- `nrow()` gives the number of rows of a dataset/matrix
- `dim()` gives the dimension

```
nrow(derby)
```

```
## [1] 145
```

```
dim(derby)
```

```
## [1] 145   8
```

4. Print the 10th row of the data table. What are the values of the variables (i.e. columns) for this row?

```
derby[10, ]
```

```
##    Year   Winner Surface Condition Starters PolePosition   Time    Speed
## 10 1884 Buchanan    dirt         0        9            8 160.25 33.69735
```

5. What are the values of the variables for rows 2 and 27?

```
derby[2, ]
```

```
##   Year  Winner Surface Condition Starters PolePosition   Time    Speed
## 2 1876 Vagrant    dirt         1       11            4 158.35 34.10167
```

```
derby[27, ]
```

```
##    Year      Winner Surface Condition Starters PolePosition   Time    Speed
## 27 1901 HisEminence    dirt         1        5            3 127.75 35.22505
```

6. What are values of the first 10 Speeds? (Speed is the 8th row.)

```
# There are several ways to approach this. Here are two:
derby[1:10, 8]
```

```
##  [1] 34.23138 34.10167 34.17722 34.34022 34.39490 34.28571 33.75000 33.69735
##  [9] 33.12883 33.69735
```

```
derby$Speed[1:10]
```

```
##  [1] 34.23138 34.10167 34.17722 34.34022 34.39490 34.28571 33.75000 33.69735
##  [9] 33.12883 33.69735
```

7.What is the range (minimum to maximum) of Speeds?

```
range(derby$Speed) # there are many built in R functions available
```

```
## [1] 31.34978 37.68844
```

```
diff(range(derby$Speed))
```

```
## [1] 6.33866
```

8. Find the mean and median Speed.

```
mean(derby$Speed)
```

```
## [1] 35.89549
```

```
median(derby$Speed)
```

```
## [1] 36.29032
```

```
# another way:
summary(derby$Speed)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   31.35   35.02   36.29   35.90   36.82   37.69
```

9. Find the variance, standard deviation, and IQR of Speeds.

```
var(derby$Speed)
```

```
## [1] 1.435286
```

```
sd(derby$Speed)
```

```
## [1] 1.198034
```

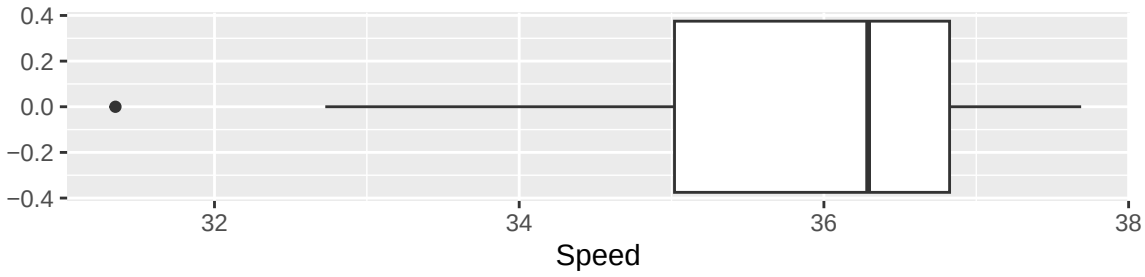
```
IQR(derby$Speed)
```

```
## [1] 1.805422
```

10. Generate a boxplot of the Speeds.

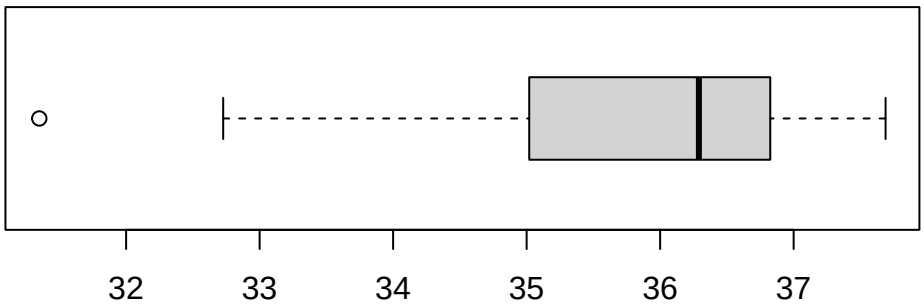
Here I demonstrate the `ggplot()` function from the `ggplot2` library...

```
ggplot(derby) + geom_boxplot(aes(Speed))
```



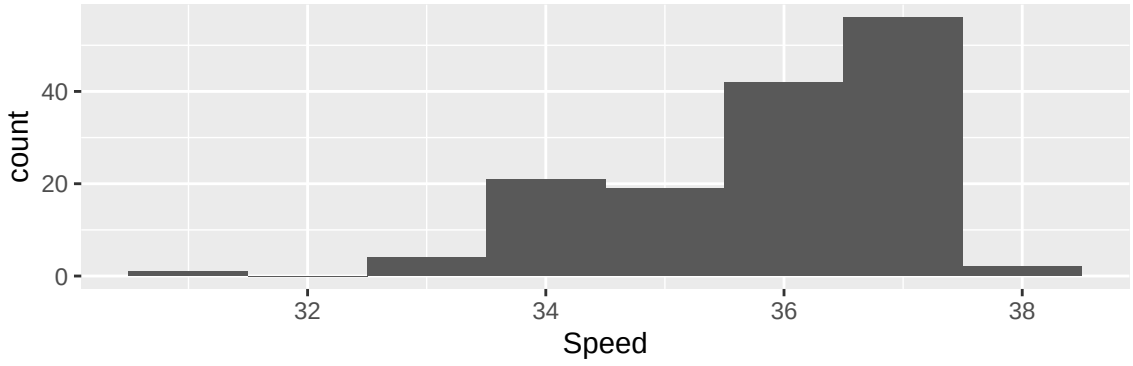
. . . and the base R boxplot() function.

```
boxplot(derby$Speed,horizontal=T)
```

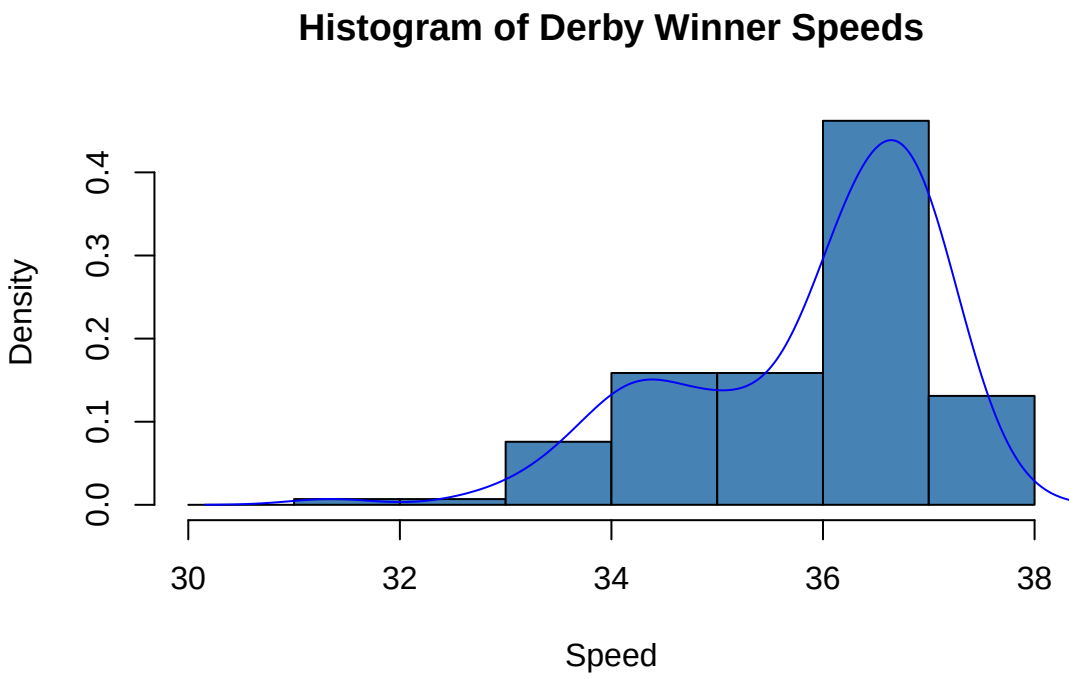


11. Generate a histogram of the Speed values. Describe the distribution of Speed.

```
ggplot(derby) + geom_histogram(aes(Speed),binwidth=1)
```



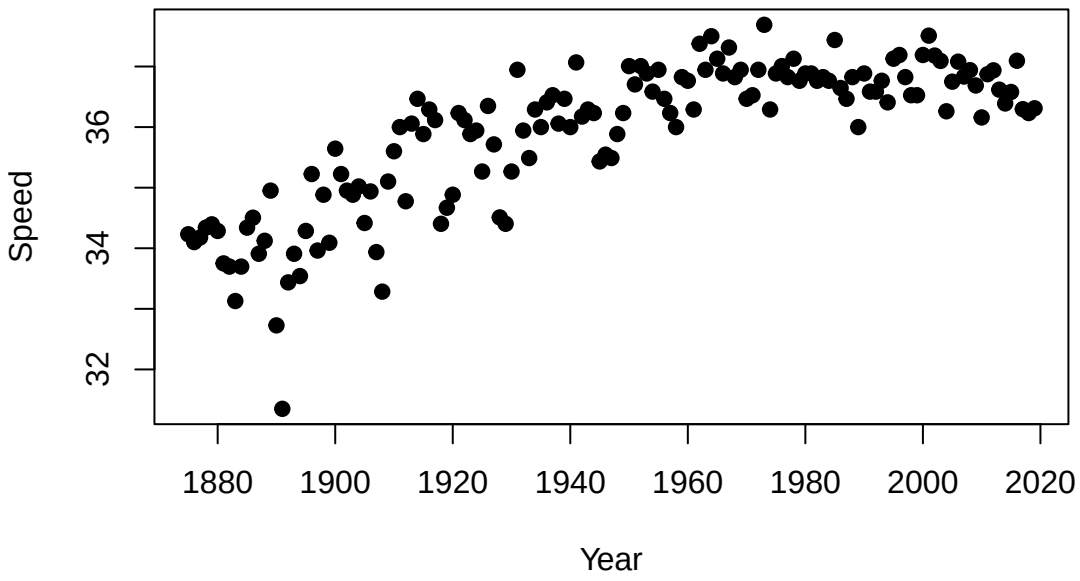

```
hist(derby$Speed,breaks=seq(30,38,1),prob=TRUE,main = "Histogram of Derby Winner Speeds",xlab="Speed",c
lines(density(derby$Speed),col="blue")
```



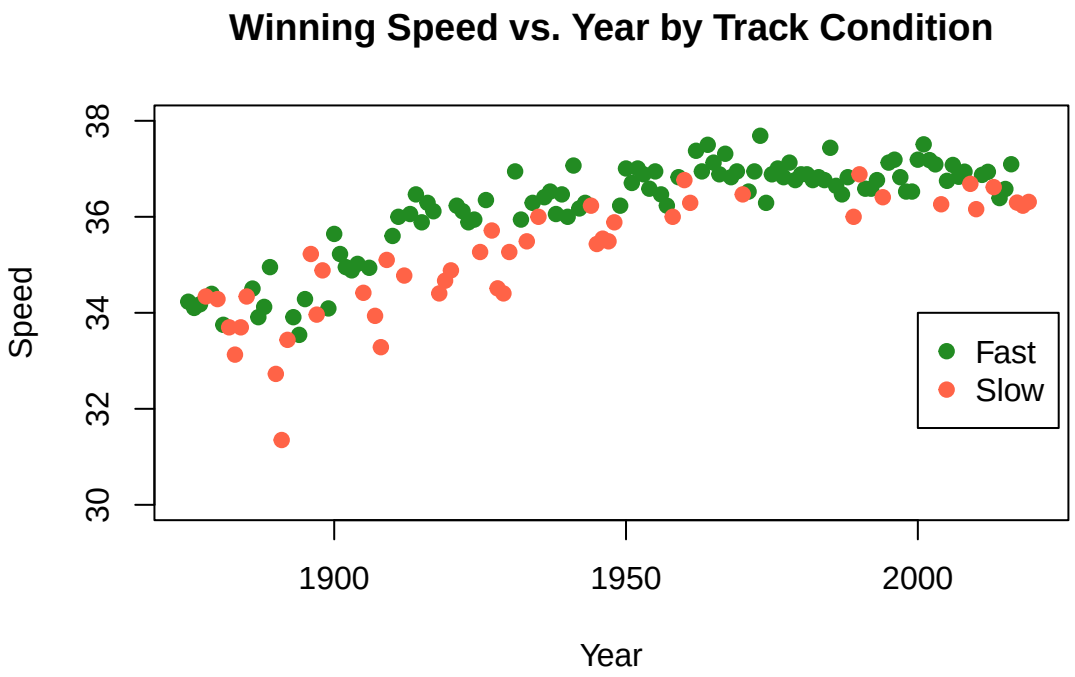
12. Make a scatterplot of the winning speed vs. year, coding separately by track condition and discuss whether/how the association changes across the track conditions and any other features of the plot you find interesting.

Base R scatterplot without including track condition

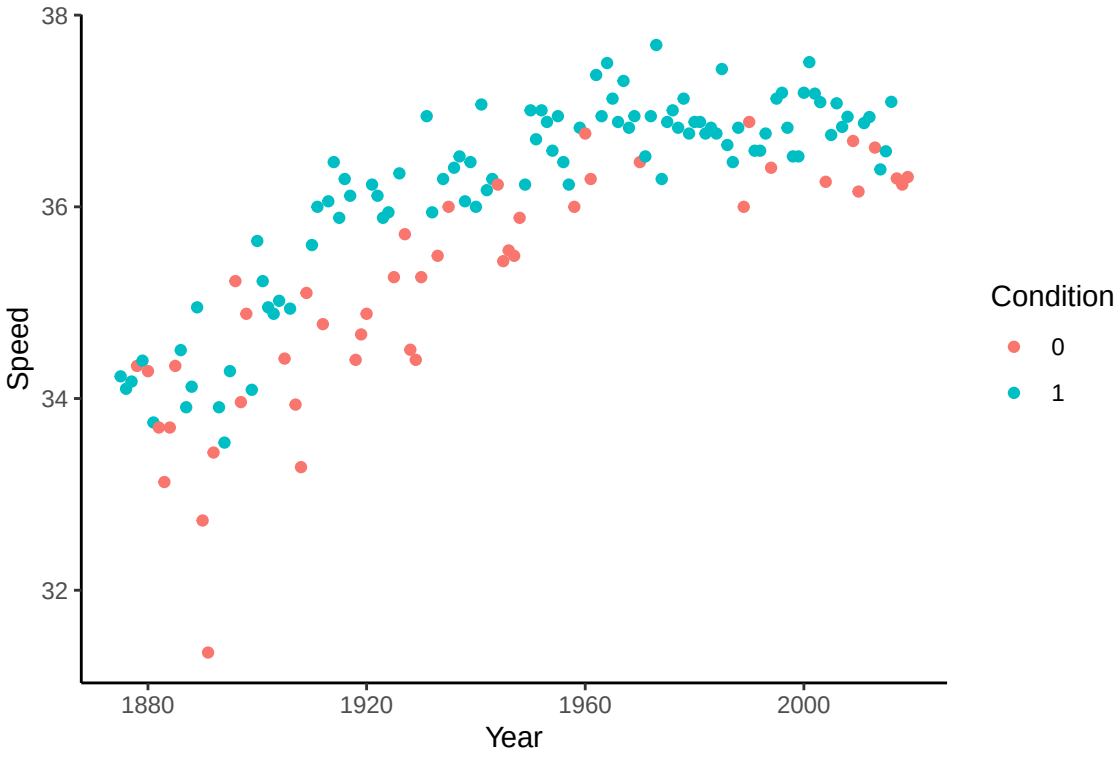
```
plot(derby$Year,derby$Speed,pch=19,xlab='Year',ylab='Speed')
```




```
# Separated out the fast vs. slow track conditions
# (there are many ways to do this!)
plot(derby$Year[derby$Condition==1],derby$Speed[derby$Condition==1],pch=19,
     ylim=c(30,38),xlim=c(1875,2020), xlab='Year',ylab='Speed',col='forestgreen',
     main = 'Winning Speed vs. Year by Track Condition') # selecting Condition = 1 (Fast)
points(derby$Year[derby$Condition==0],derby$Speed[derby$Condition==0],pch=19,
       col='tomato') # selecting Condition = 0 (Slow)
legtxt <- c("Fast","Slow")
legend(2000,34,legtxt,col=c("forestgreen","tomato"),pch=19)
```




```
# An alternative is to use ggplot()
# library(ggplot2) # Already loaded the package earlier.
p <- ggplot(derby, (aes(x=Year, y=Speed, color=Condition))) + theme_classic() +
  ylab("Speed") + xlab("Year")
p + geom_point()
```



13. What R tips or tricks do you have?

1. Write your code in an R script, or R Markdown, or Quarto file so you can save it and reproduce your analyses.
2. When you are finished with an assignment, start a clean session of R and make sure your R code runs without error.
3. Keep the .Rmd file and data file in the same folder. Or set your working directory, or declare file path.
4. Load important libraries. Use `install.packages()` once to grab the library package from a repository. Then, in any R session in which you need to access functions from that package, load the package in the session with the `library()` function.
5. The `$` operator is how you access variables within a data frame. Some like to use the `attach()` function but pay close attention to your variables when you load multiple datasets in a session.
6. Line plots require your variable to be sorted according to the ordering of the x-axis variable. The `order()` or `sort()` functions can help.
7. *KNIT frequently!* (For beginners, once every 1 - 3 chunks.)
8. Knit to HTML while working. Knit to Word for editing. Be sure to "Save as . . . " the word file.

Other important functions to explore:

9. data.frame(), matrix(), vector(), cbind(), rbind(), c()
10. seq(), sample()
11. class(), factor(), numeric()
12. pnorm(), rnorm(), dnorm(), and other probability distributions
13. help(), ??

Some resources:

14. If you use the tidyverse, then our book's labs have been implemented using tidymodels: https://emilhvitfeldt.github.io/ISLR-tidymodels-labs/index.html
15. Another resource: https://bcheggeseth.github.io/253_fall_2021/r-cheatsheet.html#real-data-example-regression

16. R Colors: http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

Example 2. Modeling A Nonlinear Relationship between Speed and Year. (Flexible and inflexible models.)

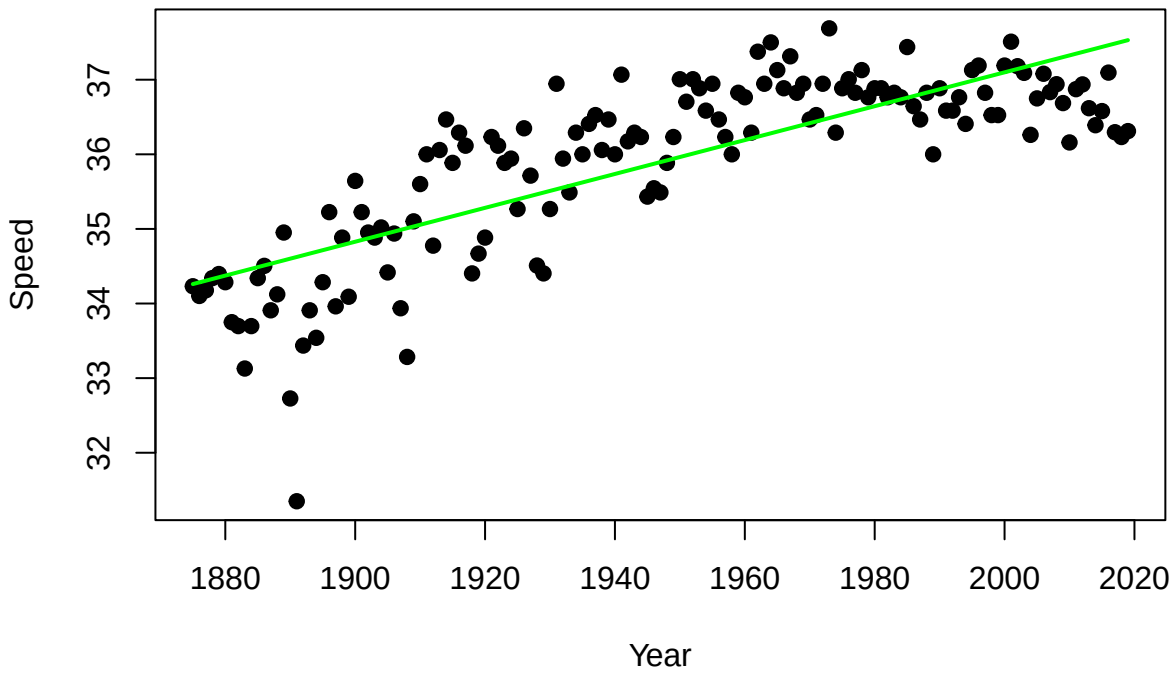
Simple linear regression model: (**Inflexible/Restrictive**)

```
# fit simple linear regression model
m1 <- lm(Speed~Year,data=derby)
summary(m1)
```

```
##
## Call:
## lm(formula = Speed ~ Year, data = derby)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -3.2742 -0.4291  0.0433  0.5294  1.4136
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.311023   2.811593  -2.956  0.00365 **
## Year         0.022705   0.001444  15.727  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7277 on 143 degrees of freedom
## Multiple R-squared:  0.6336, Adjusted R-squared:  0.6311
## F-statistic: 247.3 on 1 and 143 DF,  p-value: < 2.2e-16
```

Scatterplot of observed Speed vs. Year data and fitted line:

```
plot(derby$Year,derby$Speed,pch=19,xlab='Year',ylab='Speed')
lines(derby$Year,m1$fitted.values,col='green',lwd=2)
```



Polynomial fits:

```
m2 <- lm(Speed~Year+I(Year^2),data=derby)
summary(m2)
```

```
##
## Call:
## lm(formula = Speed ~ Year + I(Year^2), data = derby)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.91162 -0.30951  0.03081  0.40055  1.14683
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.001e+03  1.207e+02  -8.294 7.62e-14 ***
## Year         1.043e+00  1.240e-01   8.409 3.96e-14 ***
## I(Year^2)   -2.620e-04  3.185e-05  -8.226 1.12e-13 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.6009 on 142 degrees of freedom
## Multiple R-squared:  0.7519, Adjusted R-squared:  0.7484
## F-statistic: 215.2 on 2 and 142 DF,  p-value: < 2.2e-16
```

```
m3 <- lm(Speed~Year+I(Year^2)+I(Year^3),data=derby)
summary(m3)
```

```
##
## Call:
## lm(formula = Speed ~ Year + I(Year^2) + I(Year^3), data = derby)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.90991 -0.34460  0.01901  0.36104  1.26635
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.194e+04  6.321e+03   1.888   0.0610 .
## Year        -1.890e+01  9.743e+00  -1.940   0.0544 .
## I(Year^2)    9.985e-03  5.005e-03   1.995   0.0480 *
## I(Year^3)   -1.754e-06  8.569e-07  -2.047   0.0425 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5943 on 141 degrees of freedom
## Multiple R-squared:  0.759,  Adjusted R-squared:  0.7539
## F-statistic: 148.1 on 3 and 141 DF,  p-value: < 2.2e-16
```

```
m16 <- lm(Speed~poly(Year,16),data=derby)
summary(m16)
```

```
##
## Call:
## lm(formula = Speed ~ poly(Year, 16), data = derby)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.3569 -0.2929  0.0235  0.3213  1.3217
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)       35.89549    0.04858 738.928  < 2e-16 ***
## poly(Year, 16)1   11.44383    0.58495  19.564  < 2e-16 ***
## poly(Year, 16)2   -4.94360    0.58495  -8.451  5.5e-14 ***
## poly(Year, 16)3   -1.21667    0.58495  -2.080   0.0395 *
## poly(Year, 16)4    0.98174    0.58495   1.678   0.0957 .
## poly(Year, 16)5   -1.05755    0.58495  -1.808   0.0730 .
## poly(Year, 16)6    0.89311    0.58495   1.527   0.1293
## poly(Year, 16)7   -1.21606    0.58495  -2.079   0.0396 *
## poly(Year, 16)8   -0.53677    0.58495  -0.918   0.3605
## poly(Year, 16)9    0.88364    0.58495   1.511   0.1334
## poly(Year, 16)10 -0.16690    0.58495  -0.285   0.7759
## poly(Year, 16)11  0.39532    0.58495   0.676   0.5004
## poly(Year, 16)12 -0.30359    0.58495  -0.519   0.6047
## poly(Year, 16)13 -0.49809    0.58495  -0.852   0.3961
## poly(Year, 16)14 -0.04964    0.58495  -0.085   0.9325
## poly(Year, 16)15 -0.13695    0.58495  -0.234   0.8153
## poly(Year, 16)16  0.17140    0.58495   0.293   0.7700
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.585 on 128 degrees of freedom
```

```
## Multiple R-squared:  0.7881, Adjusted R-squared:  0.7616
## F-statistic: 29.75 on 16 and 128 DF,  p-value: < 2.2e-16
```

Smoothing methods (natural splines): (**Flexible**, with various df)

```
#install.packages(splines)
library(splines)
# natural spline models
# degrees of freedom = df = 20
mns20 <- lm(Speed~ns(Year,20),data=derby)
summary(mns20)
```

```
##
## Call:
## lm(formula = Speed ~ ns(Year, 20), data = derby)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.19071 -0.31156  0.03233  0.29300  1.55647
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     34.0951     0.3769  90.473  < 2e-16 ***
## ns(Year, 20)1   -1.3353     0.5172  -2.582 0.010992 *
## ns(Year, 20)2    0.8383     0.6440   1.302 0.195415
## ns(Year, 20)3    0.3686     0.5831   0.632 0.528522
## ns(Year, 20)4    1.2486     0.6175   2.022 0.045311 *
## ns(Year, 20)5    1.8280     0.5996   3.049 0.002811 **
## ns(Year, 20)6    1.2300     0.6093   2.019 0.045689 *
## ns(Year, 20)7    1.6318     0.6042   2.701 0.007884 **
## ns(Year, 20)8    2.7138     0.6069   4.471 1.73e-05 ***
## ns(Year, 20)9    1.3659     0.6055   2.256 0.025820 *
## ns(Year, 20)10   3.0234     0.6063   4.987 2.02e-06 ***
## ns(Year, 20)11   2.4257     0.6058   4.004 0.000107 ***
## ns(Year, 20)12   3.1897     0.6061   5.263 6.03e-07 ***
## ns(Year, 20)13   2.4918     0.6059   4.113 7.07e-05 ***
## ns(Year, 20)14   3.0933     0.6059   5.106 1.21e-06 ***
## ns(Year, 20)15   2.1262     0.6055   3.512 0.000622 ***
## ns(Year, 20)16   3.0857     0.6043   5.107 1.20e-06 ***
## ns(Year, 20)17   2.7142     0.5999   4.525 1.40e-05 ***
## ns(Year, 20)18   2.5533     0.5030   5.076 1.38e-06 ***
## ns(Year, 20)19   2.6797     0.9681   2.768 0.006503 **
## ns(Year, 20)20   2.0540     0.4363   4.708 6.58e-06 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5715 on 124 degrees of freedom
## Multiple R-squared:  0.8041, Adjusted R-squared:  0.7725
## F-statistic: 25.44 on 20 and 124 DF,  p-value: < 2.2e-16
```

```
# df = 50
mns50 <- lm(Speed~ns(Year,50),data=derby)
summary(mns50)
```

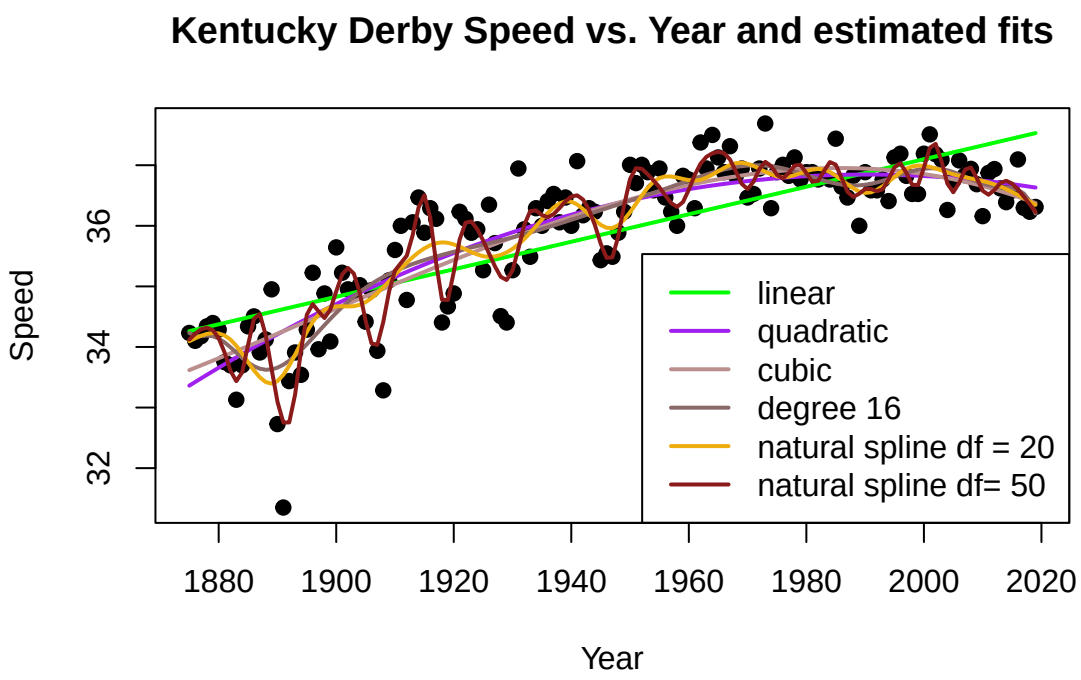
```
##
## Call:
```

```
## lm(formula = Speed ~ ns(Year, 50), data = derby)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.40306 -0.19337  0.00976  0.21547  1.32176
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    34.12141    0.42881  79.573  < 2e-16 ***
## ns(Year, 50)1   0.02609    0.67741   0.039 0.969362
## ns(Year, 50)2  -1.26656    0.80584  -1.572 0.119375
## ns(Year, 50)3   1.20349    0.74940   1.606 0.111638
## ns(Year, 50)4  -0.86601    0.78270  -1.106 0.271358
## ns(Year, 50)5  -2.02193    0.76538  -2.642 0.009661 **
## ns(Year, 50)6   1.23997    0.77430   1.601 0.112640
## ns(Year, 50)7  -0.10414    0.76909  -0.135 0.892576
## ns(Year, 50)8   1.31706    0.77181   1.706 0.091225 .
## ns(Year, 50)9   1.17401    0.77075   1.523 0.131063
## ns(Year, 50)10 -0.86440    0.77193  -1.120 0.265655
## ns(Year, 50)11  1.49648    0.77180   1.939 0.055507 .
## ns(Year, 50)12  1.03005    0.77202   1.334 0.185352
## ns(Year, 50)13  3.36690    0.77163   4.363 3.28e-05 ***
## ns(Year, 50)14 -0.48276    0.77127  -0.626 0.532876
## ns(Year, 50)15  2.17393    0.77088   2.820 0.005856 **
## ns(Year, 50)16  1.89644    0.77084   2.460 0.015711 *
## ns(Year, 50)17  1.21887    0.77110   1.581 0.117311
## ns(Year, 50)18  0.69064    0.77154   0.895 0.372992
## ns(Year, 50)19  2.44750    0.77188   3.171 0.002053 **
## ns(Year, 50)20  1.87218    0.77198   2.425 0.017211 *
## ns(Year, 50)21  2.22115    0.77178   2.878 0.004955 **
## ns(Year, 50)22  2.51485    0.77139   3.260 0.001551 **
## ns(Year, 50)23  1.96388    0.77098   2.547 0.012481 *
## ns(Year, 50)24  0.78693    0.77081   1.021 0.309917
## ns(Year, 50)25  3.03516    0.77098   3.937 0.000158 ***
## ns(Year, 50)26  2.75425    0.77139   3.571 0.000564 ***
## ns(Year, 50)27  2.46497    0.77178   3.194 0.001911 **
## ns(Year, 50)28  1.96878    0.77198   2.550 0.012379 *
## ns(Year, 50)29  2.93129    0.77188   3.798 0.000259 ***
## ns(Year, 50)30  3.11733    0.77153   4.040 0.000109 ***
## ns(Year, 50)31  3.11038    0.77111   4.034 0.000112 ***
## ns(Year, 50)32  2.13422    0.77083   2.769 0.006780 **
## ns(Year, 50)33  3.27195    0.77089   4.244 5.14e-05 ***
## ns(Year, 50)34  2.36996    0.77124   3.073 0.002774 **
## ns(Year, 50)35  3.19609    0.77167   4.142 7.52e-05 ***
## ns(Year, 50)36  2.24991    0.77194   2.915 0.004451 **
## ns(Year, 50)37  3.32937    0.77194   4.313 3.97e-05 ***
## ns(Year, 50)38  2.08891    0.77167   2.707 0.008065 **
## ns(Year, 50)39  2.65616    0.77124   3.444 0.000858 ***
## ns(Year, 50)40  2.24365    0.77089   2.910 0.004506 **
## ns(Year, 50)41  3.31110    0.77082   4.296 4.24e-05 ***
## ns(Year, 50)42  1.98626    0.77108   2.576 0.011554 *
## ns(Year, 50)43  3.92065    0.77143   5.082 1.89e-06 ***
## ns(Year, 50)44  1.86236    0.77152   2.414 0.017721 *
## ns(Year, 50)45  3.29543    0.77072   4.276 4.57e-05 ***
```

```
## ns(Year, 50)46  2.10312    0.76741   2.741 0.007341 **
## ns(Year, 50)47  2.75629    0.75602   3.646 0.000437 ***
## ns(Year, 50)48  2.40057    0.63862   3.759 0.000296 ***
## ns(Year, 50)49  2.47595    1.12220   2.206 0.029796 *
## ns(Year, 50)50  1.94318    0.52148   3.726 0.000332 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.4884 on 94 degrees of freedom
## Multiple R-squared:  0.8915, Adjusted R-squared:  0.8338
## F-statistic: 15.45 on 50 and 94 DF,  p-value: < 2.2e-16
```

Scatterplot of observed Speed vs. Year data and fitted curves:

```
plot(derby$Year,derby$Speed,pch=19,xlab='Year',ylab='Speed',
     main='Kentucky Derby Speed vs. Year and estimated fits')
lines(derby$Year,m1$fitted.values,col='green',lwd=2)
lines(derby$Year,m2$fitted.values,col='purple',lwd=2)
lines(derby$Year,m3$fitted.values,col='rosybrown',lwd=2)
lines(derby$Year,m16$fitted.values,col='rosybrown4',lwd=2)
lines(derby$Year,mns20$fitted.values,col='darkgoldenrod2',lwd=2)
lines(derby$Year,mns50$fitted.values,col='firebrick4',lwd=2)
legtxt <- c("linear","quadratic","cubic","degree 16","natural spline df = 20",
            "natural spline df= 50")
legend("bottomright",legtxt,col=c("green","purple","rosybrown","rosybrown4",
                                  "darkgoldenrod2","firebrick4"),lwd=2,lty=1)
```



Example 3. Assessing Model Accuracy using MSE.

Recall that interest in estimating the unknown function $f(x)$ where $Y = f(x) + \varepsilon$. Consider a given estimate $\hat{f}$ and predictor(s) x such that $\hat{Y} = \hat{f}(x)$. Assume $\hat{f}$ and x are fixed (so that variability only comes from ε). Then,

$E[(Y - \hat{Y})^2]$ = Expected squared difference between the observed (i.e. actual) Y and the predicted Y (i.e. $\hat{Y}$)

In a regression setting, the most commonly used measure to estimate $E[(Y - \hat{Y})^2]$ is the (observed) mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}[y_i - \hat{f}(x_i)]^2$$

.

Note that the above MSE is slightly different from the MSE in the ANOVA table (linear regression, design of experiments) in its denominator.

$$MSE(\text{in ANOVA}) = \frac{1}{df_{error}}\sum_{i=1}^{n}[y_i - \hat{f}(x_i)]^2$$

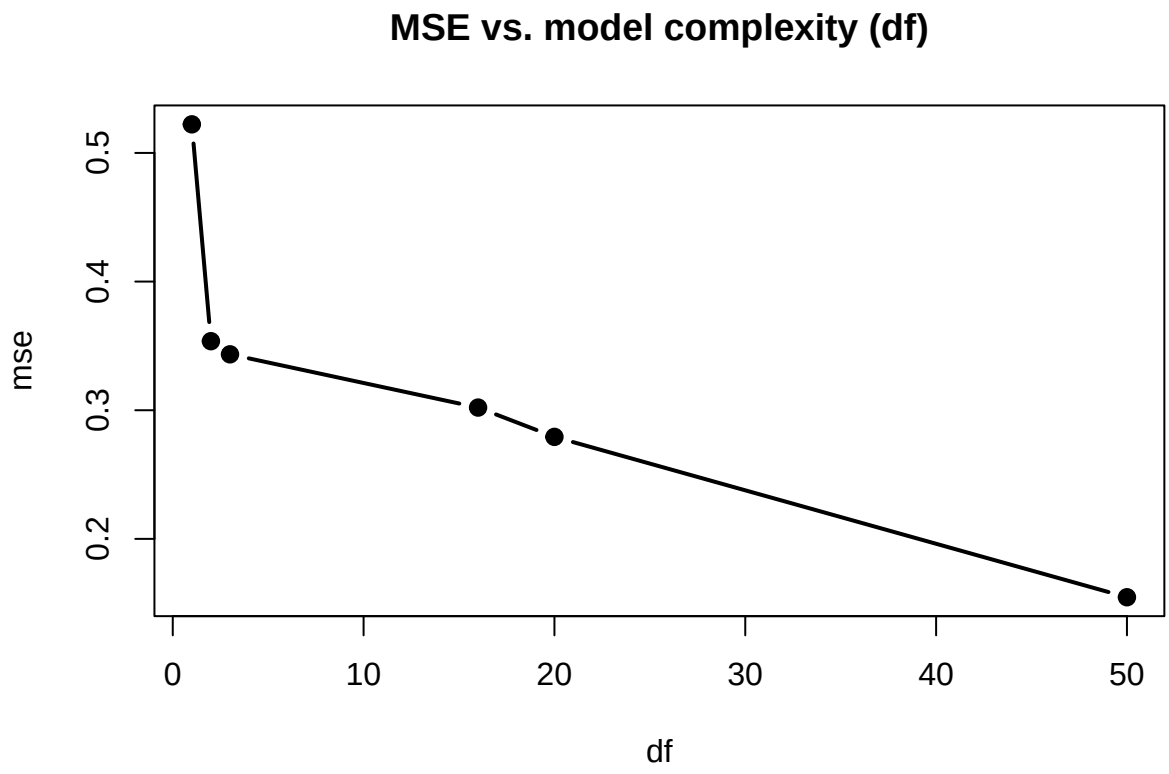
```
sse1 <- sum((derby$Speed-m1$fitted.values)^2)
sse2 <- sum((derby$Speed-m2$fitted.values)^2)
sse3 <- sum((derby$Speed-m3$fitted.values)^2)
sse16 <- sum((derby$Speed-m16$fitted.values)^2)
sse_ns20 <- sum((derby$Speed-mns20$fitted.values)^2)
sse_ns50 <- sum((derby$Speed-mns50$fitted.values)^2)

mse1 <- mean((derby$Speed-m1$fitted.values)^2)
mse2 <- mean((derby$Speed-m2$fitted.values)^2)
mse3 <- mean((derby$Speed-m3$fitted.values)^2)
mse16 <- mean((derby$Speed-m16$fitted.values)^2)
mse_ns20 <- mean((derby$Speed-mns20$fitted.values)^2)
mse_ns50 <- mean((derby$Speed-mns50$fitted.values)^2)

mse.out <- data.frame(df=c(1,2,3,16,20,50),
                      sse=rbind(sse1,sse2,sse3,sse16,sse_ns20,sse_ns50),
                      mse=rbind(mse1,mse2,mse3,mse16,mse_ns20,mse_ns50))
row.names(mse.out) <- c("linear","quadratic","cubic","degree 16",
                        "natural spline df = 20", "natural spline df= 50")
mse.out
```

```
##                          df      sse       mse
## linear                    1 75.72000 0.5222069
## quadratic                 2 51.28078 0.3536606
## cubic                     3 49.80050 0.3434518
## degree 16                16 43.79790 0.3020545
## natural spline df = 20   20 40.49661 0.2792870
## natural spline df= 50    50 22.42200 0.1546345
```

```
plot(mse.out$df, mse.out$mse,xlab='df',ylab='mse',pch=19,type='b',lwd=2,
     main='MSE vs. model complexity (df)')
```



Does it mean we should keep increasing the model flexibility?

NO!

- Training MSE typically decreases as the model becomes more flexible.
- Test MSE will be small when the model is *correct*.

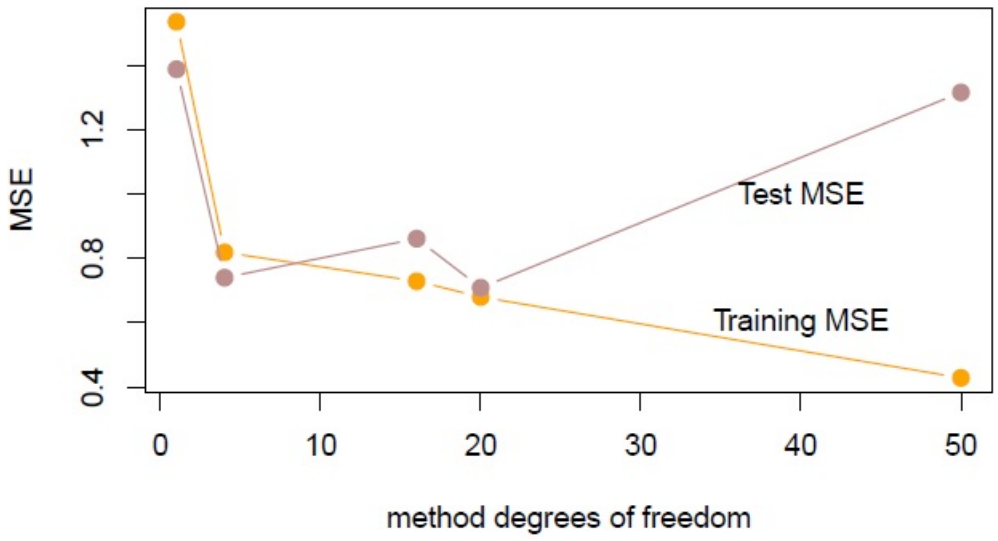


Figure 1: Training and test MSE (from a simulated data set)