All_lectures

Code

Math operators found at https://rpruim.github.io/s341/S19/from-class/MathinRmd.html (https://rpruim.github.io/s341/S19/from-class/MathinRmd.html)

Hide

```
?Matrix
library(Matrix)
library(tidymodels)
library(knitr)
library(dplyr)
library(tidyverse)
library(tidyr)
library(ggplot2)
library(readr)
library(caret)
library(e1071)
library(caTools)
library(rsample)
```

Lecture 3 PH + in class

In class assignment

Problem 1.1/1.2

Hide

```
boston <- read.delim("boston_corrected.txt", skip = 9, header = TRUE)
dim(boston)
```

```
[1] 506  21
```

Hide

```
sum(is.na(boston))
```

```
[1] 0
```

problem 3

%>% is a piping function. Rather than specifying the variable i want to target for all operations I have already prompted the updated variable. Syntax boston <- select(boston,-“OBS.”,-“TOWN”,-“TOWN.”,-“TRACT”,-“LON”,-“LAT”,-“MEDV”) would give same result. Piping is useful when you wish to to additional variable updates in a row.

Hide

```
boston <- boston %>%
  select(-"OBS.",-"TOWN",-"TOWN.",-"TRACT",-"LON",-"LAT",-"MEDV")
```

problem 4

remember to save variable change to the updated variable using <-

Be aware that in this example the logic is:

Names(boston) - the columns of df boston -, needs to be set to lower. We need to both save the lowercase names to our dataframe and specify we want them lower, as such names(boston)) is called twice.

Hide

```
names(boston) <- tolower(names(boston))
```

problem 5

Hide

```
boston <- rename(boston,medv=cmedv)
head(boston)
```

	medv <dbl>	crim <dbl>	zn <dbl>	indus <dbl>	chas <int>	nox <dbl>	rm <dbl>	age <dbl>	dis <dbl>
1	24.0	0.00632	18	2.31	0	0.538	6.575	65.2	4.0900
2	21.6	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671
3	34.7	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671
4	33.4	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622
5	36.2	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622
6	28.7	0.02985	0	2.18	0	0.458	6.430	58.7	6.0622

6 rows | 1-10 of 14 columns

problem 6

I'm using the readr write_csv functionality. It's smart and knows what to do regarding headers etc already. If you were to use base R I would do the function as follows

write.csv(boston, file = "BostonBI.csv", na = "NA") specifying of additional parameters

Hide

```
write_csv(boston, "BostonBI.csv")
```

Problem 2.2

Hide

```
boston1 <- boston %>%
  select(tax, medv)
  summary(boston1)
```

```
      tax             medv      
 Min.   :187.0   Min.   : 5.00  
 1st Qu.:279.0   1st Qu.:17.02  
 Median :330.0   Median :21.20  
 Mean   :408.2   Mean   :22.53  
 3rd Qu.:666.0   3rd Qu.:25.00  
 Max.   :711.0   Max.   :50.00  
```

Hide

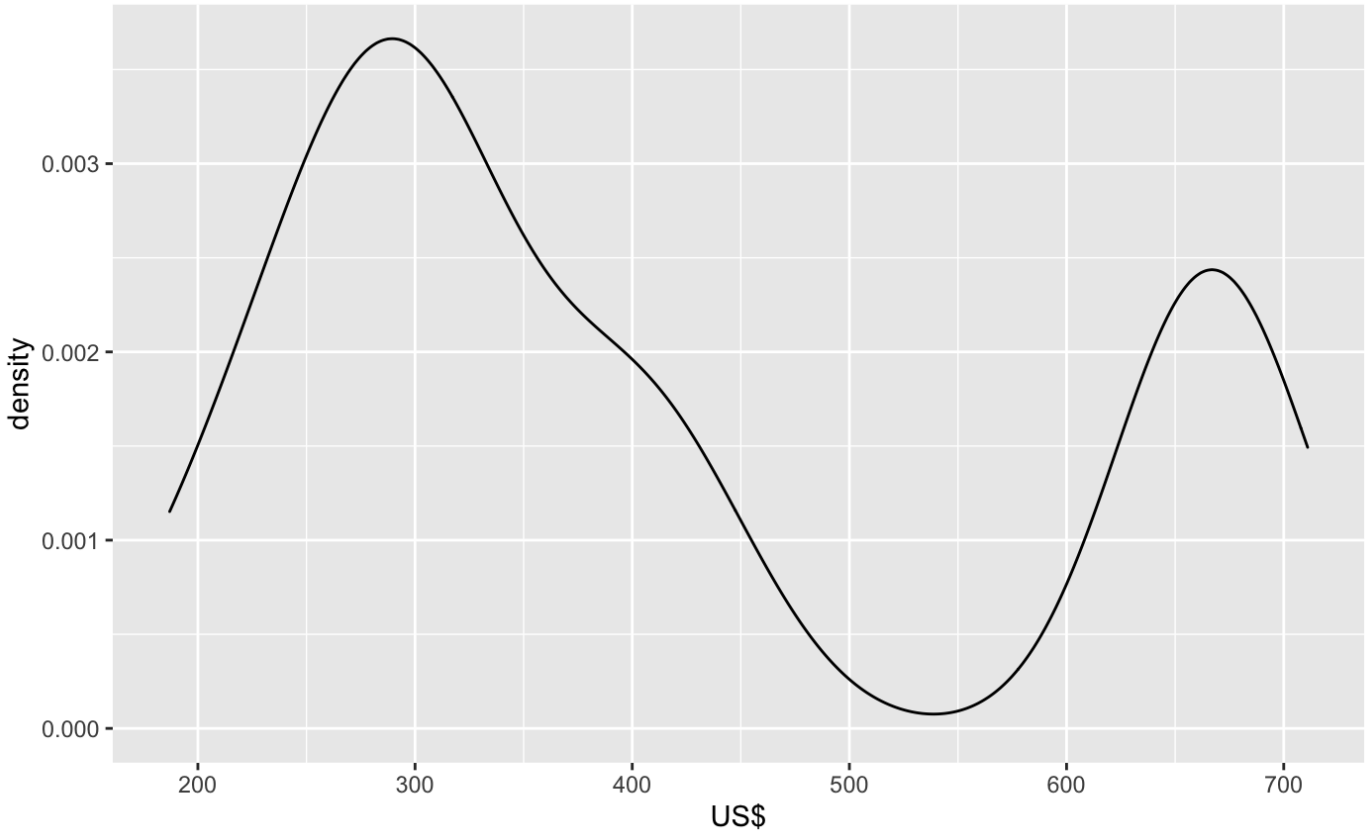
```
  cor(boston1)
```

```
            tax       medv
tax   1.0000000 -0.4719788
medv -0.4719788  1.0000000
```

Problem 3

Hide

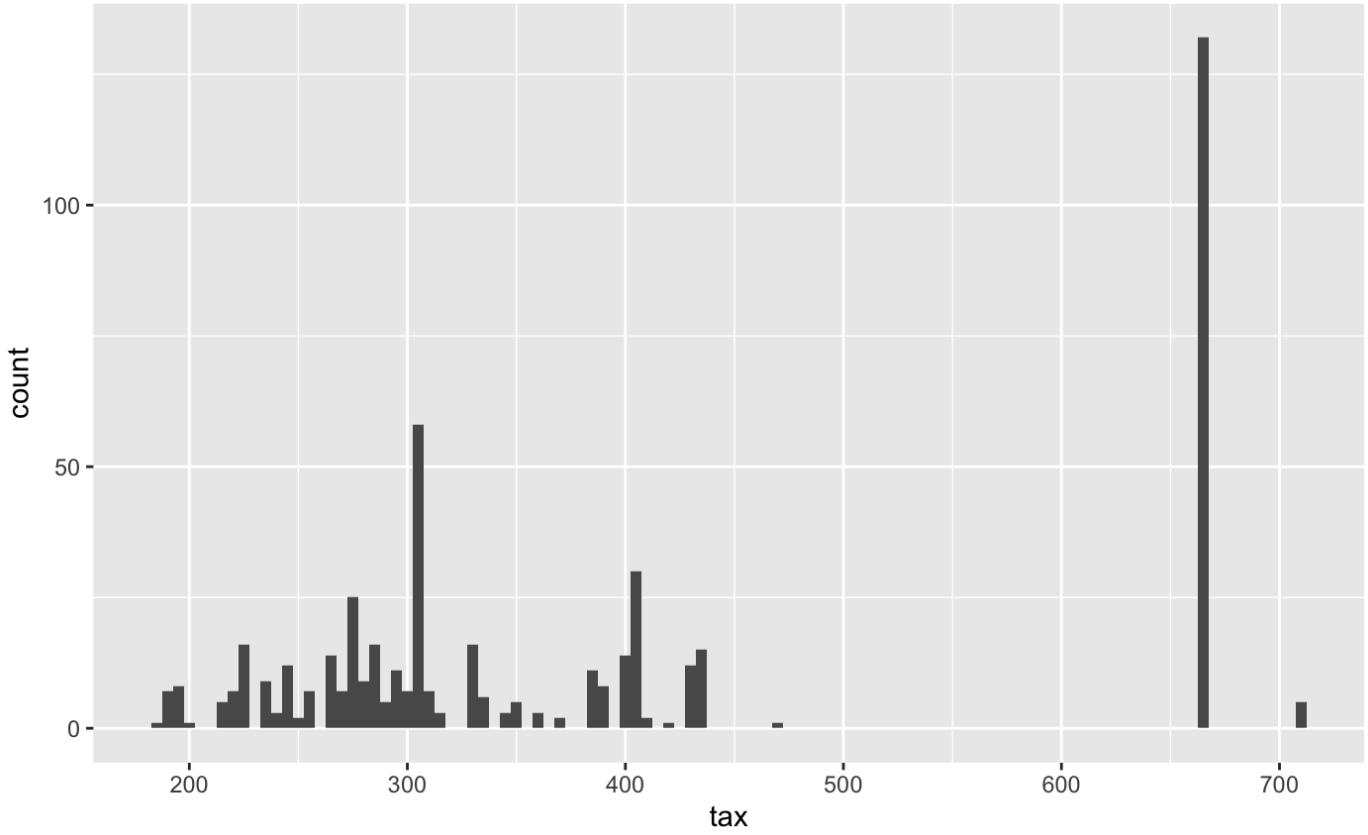
```
ggplot(boston1, aes(x=tax)) + geom_density() + labs(x="US$")
```



Problem 4

Hide

```
ggplot(boston1, aes(x=tax)) + geom_histogram(binwidth = 5)
```



problem 5

Need to attach the dataframe in order to do the factorial operations. Could probably be solved by piping? Will add to later.

First we define our string value factorial, taxlabel. Then we define the value parameters assigned to the string. If tax is less than 300, the value assigned is low, or 0. If between 300 and 600 value assigned is medium, or 1. If above 600 value assigned is high, or 3.

The factor(tax_discrete,0:2,taxlabel) defines the range spread for string values of the factorial.

Hide

```
attach(boston)
```

```
The following objects are masked from boston (pos = 3):

    age, b, chas, crim, dis, indus, lstat, medv, nox, ptratio, rad, rm, tax, zn
```

Hide

```
taxlabel <- c("low","medium","high")
tax_discrete <- 0 + (tax > 300) + (tax < 600)
tax2 <- factor(tax_discrete,0:2,taxlabel)

boston <- cbind(boston,tax2)

head(boston)
```

	medv <dbl>	crim <dbl>	zn <dbl>	indus <dbl>	chas <int>	nox <dbl>	rm <dbl>	age <dbl>	dis <dbl>
1	24.0	0.00632	18	2.31	0	0.538	6.575	65.2	4.0900
2	21.6	0.02731	0	7.07	0	0.469	6.421	78.9	4.9671

	medv <dbl>	crim <dbl>	zn <dbl>	indus <dbl>	chas <int>	nox <dbl>	rm <dbl>	age <dbl>	dis <dbl>
3	34.7	0.02729	0	7.07	0	0.469	7.185	61.1	4.9671
4	33.4	0.03237	0	2.18	0	0.458	6.998	45.8	6.0622
5	36.2	0.06905	0	2.18	0	0.458	7.147	54.2	6.0622
6	28.7	0.02985	0	2.18	0	0.458	6.430	58.7	6.0622

6 rows | 1-10 of 15 columns

Hide

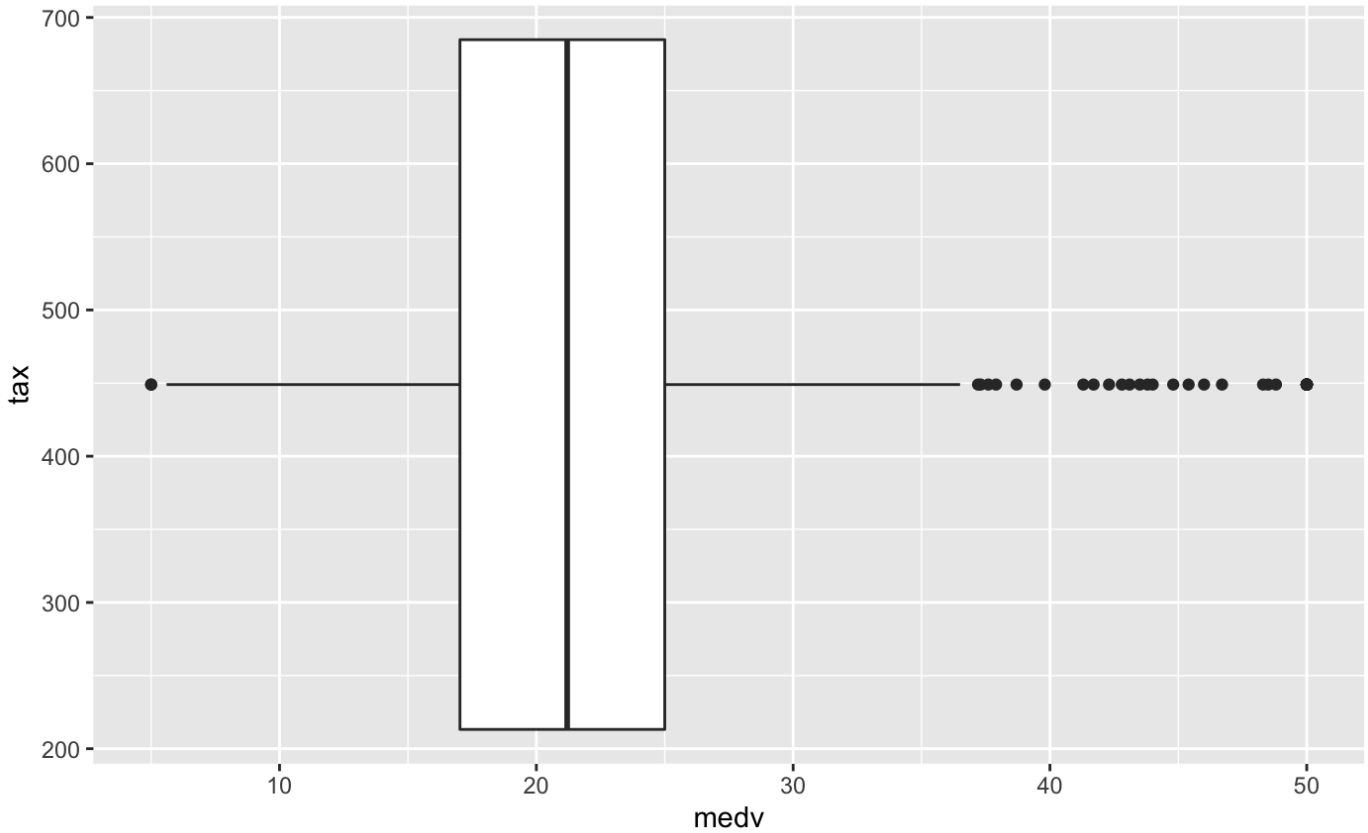
```
NA
NA
```

Problem 6

Coord_flip flips the x and y values. Used here for increased readability.

Hide

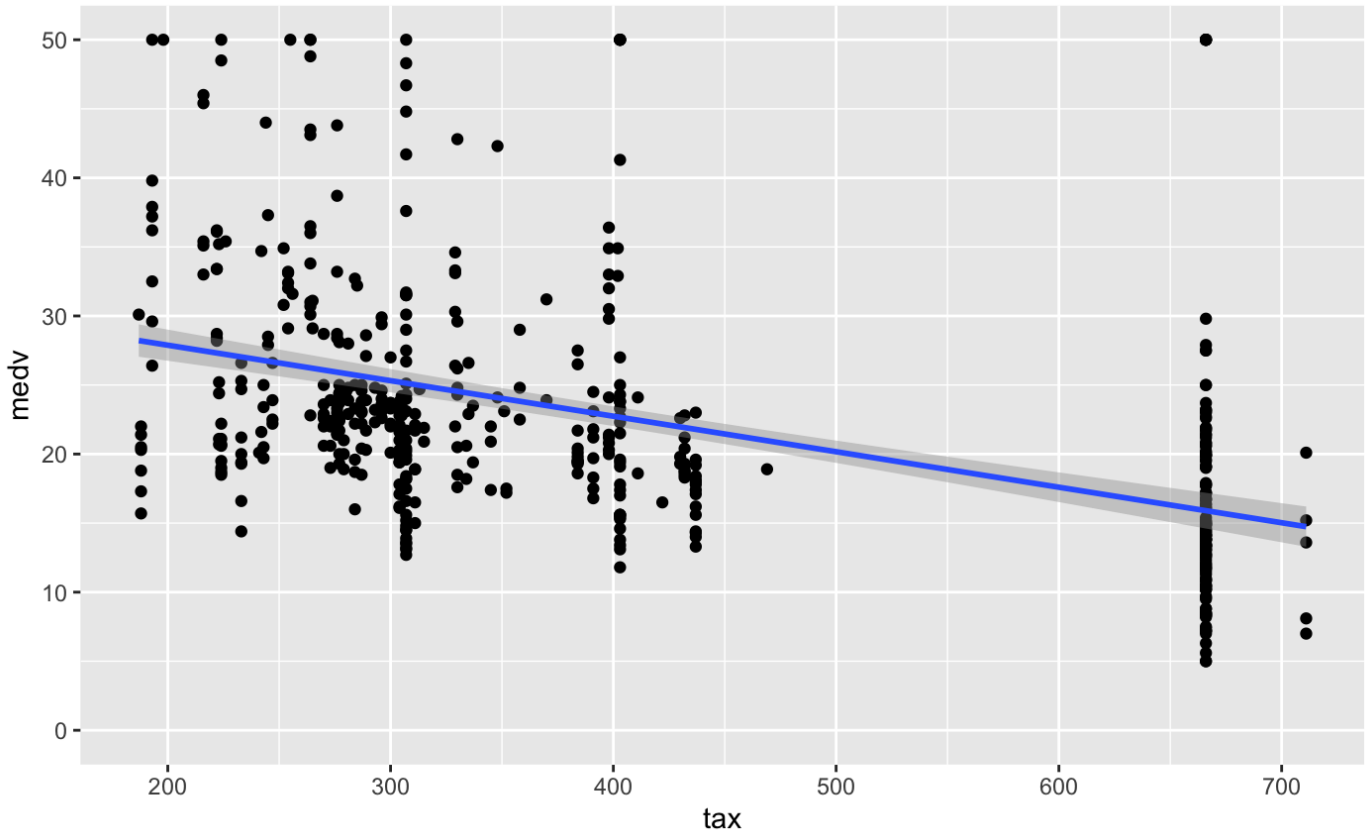
```
ggplot(boston, aes(x=tax,y=medv)) + geom_boxplot() + coord_flip()
```



###problem 7 ggplot syntax = aes = Aesthetic (udseende) geom = type of chart, point = scatter, boxplot = boxplot, histogram = histo. stat_smooth is a best fit model, you can define method of the fit by (method="x"), here "lm" is linear model fit

Hide

```
ggplot(boston, aes(x=tax,y=medv)) +
  geom_point() +
  ylim(0,50) +
  # stat_smooth()
  stat_smooth(method="lm")
```



Hide

```
NA
NA
```

Lecture 4

Predictive learnings

Input = independent variables (IV) = features = predictors = X

$$x_1, \dots x_n$$

Output = Dependent variables (DV) = response = Y

$$Y_1, \dots, y_m$$

other var. that affect Y, but those values are neither observed nor controlled (noise?)

$$Z_1, \dots Z_k$$

Matemathical model

$$(1) y_k = g_k(x1, \dots, x_n, z_1, z_L), = k = 1, K$$

y_k can be any row in our dataset

Statistical model

$$(2) y_k = f_k(x_1, \ldots, x_n) + e_k, k = 1, K$$

f_k = Function of the observed inputs

$$f_k$$

e_k = an additional random stochastic component / error term

$$e_k$$

If denoting

$$X = (x_1, x_2, \ldots, x_p)$$

then (2) becomes

$$y_k = f(X) + e_k$$

even if we find the perfect approximation of $f(X)$ we will never be able to compute for the random factor e_k

f Is used as an estimation for new y observations, which helps us understand the mechanism that is produced by the data (y) output to help intervene in the future

Example 1(slide 14) (ISL p. 16-17) is an Ordinary least squares (OLSS) regression Example 2(slide 15) (ISL, p. 16-17) OLS estimation can be viewed as a projection onto the linear space spanned by the regressors.

For predictions: - Focus on reducible errors

if:

$$E(Y - \hat{Y})^2 = E[f(X) + e - \hat{f}(X)]^2$$

Then

$$E(Y - \hat{Y})^2 = [f(X) - \hat{f}(X)]^2 + Var(e)$$

Where $[f(X) - \hat{f}(X)]^2$ is reducible and $Var(e)$is irreducible

For inference put focus on - Which predictors X associate with response y - Magnitude & direction - relationship (Linear or other) - interaction effects

In ML we're mainly focused on the predictive perspective rather than the interference - It is however possible in some situations to focus on both aspects at once

ESTIMATION OF *f*

1. Parametric
2. Non-parametric

1. Parametric

'a priori assumption" : Relating to ro denoting reasoning or knowledge which proceeds from theoretical deduction rather than observation or experience, ie. "sexuality may be a factor but it cannot be assumed a priori"

We fit the model based on our a priori assumptions. If we expect linear fit we estimate the parameters beta (Multiple linear regression. If you use ≈ you don't note the error variable e as it's an estimation)

$$y \approx ß_o + ß_1 X_1 + ß_2 X_2 + \ldots + ß_p X_p$$

where $ß_0, ß_1, ß_2$ and $ß_p$ are our estimators

For example 1 slide 23 $income \approx ß_0 + ß_1 YoE_1 + ß_2 Seniority_2$

for example 2 slide 24 (Polynominal and interaction included),(Followup after class, incorrect)
$PI \approx ß_o + ß_1 X_1 + ß_2 X_2 + \ldots + ß_p X_p$. The interaction changes the curvature of the 2-dimensional plane.

2. Non-parametric

No assumption; f is very flexible

Advantages: Predictive accuracy Disadvantages: Large number og observations is required; overfitting risk; low interpretability. Non a priori

example from class: Crumbled up paper. Imagine we have to predict the plane of the paper, it's incredibly difficult as the paper is all crambled up. Sometimes in non-parametric f can be considered a blackbox as it's borderline impossible to approximate.

will be covered in Machine Learning 2

method suitability

Hide

```
include_graphics("flexibilty vs interp.png")
```

Flexibility vs. interpretability for different methods

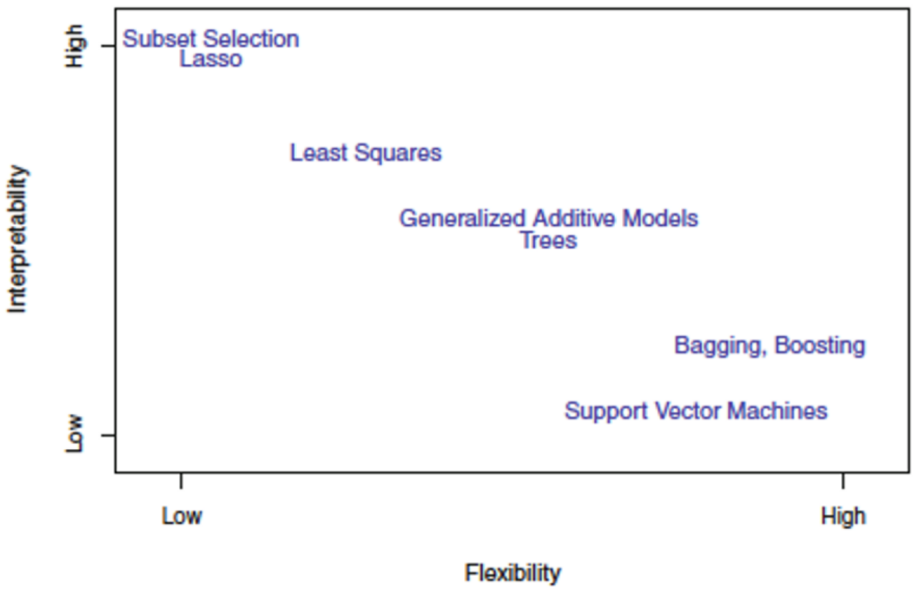


FIGURE 2.7. *A representation of the tradeoff between flexibility and interpretability, using different statistical learning methods. In general, as the flexibility of a method increases, its interpretability decreases.*

Data partitioning

Split datasets into partitions, one for training and one for testing Example: From Tutoral 1 we know that (dataset not included)

Hide

```
# library(rsample)
# set.seed(123)
# split_1  <- initial_split(df, prop = 0.6)
# train <- training(split_1)
# test <- testing(split_1)
```

For above; Load rsample library for the utilties

Seed for reproductibility - The seed specifies the point at which we would like to split the dataset. Without a seed a random number will be assigned (number assigned is not actually random but rather pseudorandom)

define variable of split = split_1, where df is our dataframe or tabel.

Prop defines where the dataset is split, 0.6 = 60/40, 0.5 would equal 50/50, etc.

training and testing are functions of the rsample library. using these functions with our split variable will automatically assign the desired split

Approaches for partitioning¨

splits will typically be done in an 80:20 fashion (prop = 0.8)

1. Random split

- Tradtional technique, simplest way. Used in the book.

2. Stratified split

- Considers target variable(y) and will try to group for it before split - if dataset has “1” and “0” varlues and you pick randomly it can split the sets poorly, here grouping them beforehand ensures data integrity for both sets

expand with knowledge from https://bradleyboehmke.github.io/HOML/process.html (https://bradleyboehmke.github.io/HOML/process.html)

Re-sampling

Hide

```
include_graphics("ML process.png")
```

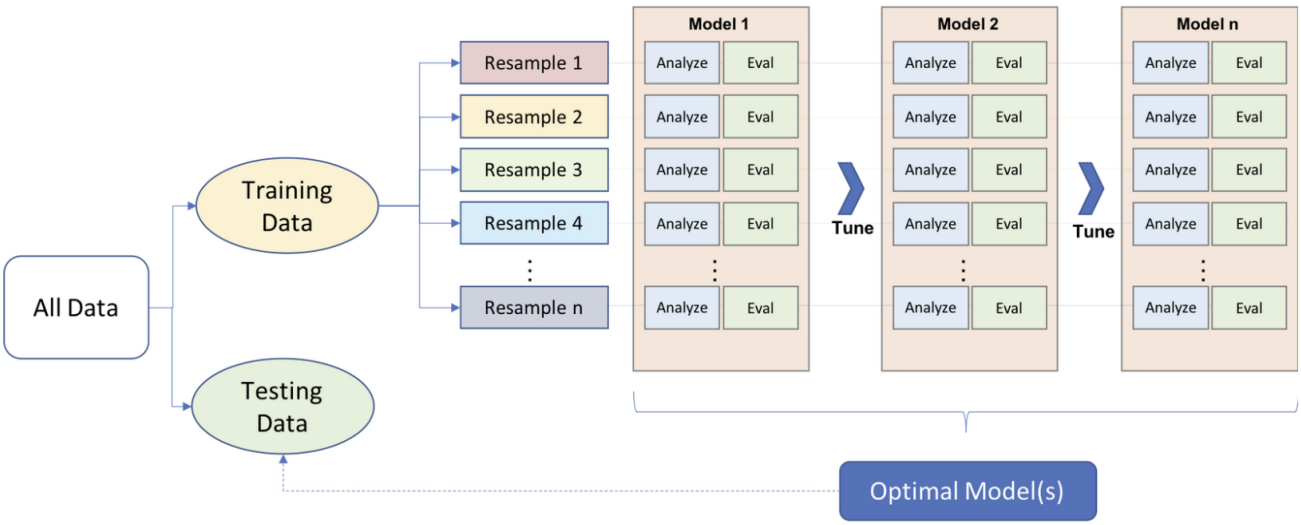


Figure 2.1: General predictive machine learning process.

- Single training data leads to inaccurate results. To avoid this we utilize re-sampliung methods

k-fold cross validation

We can force the training sets to be stratified throughout the k-fold cross validation.

Hide

```
include_graphics("k-fold method.png")
```

k-fold cross validation[1]

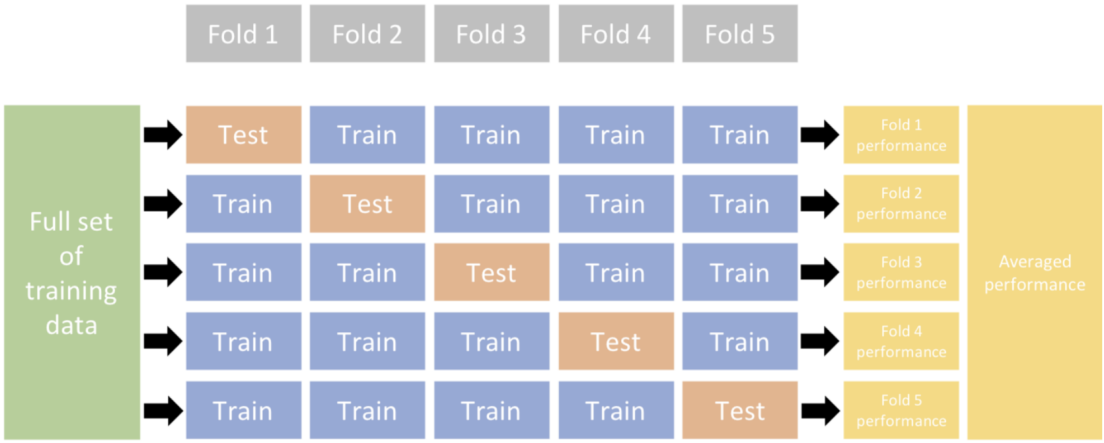


Figure 2.4: Illustration of the k-fold cross validation process.

[1] Iteratively train each model (f) on a fraction of the training data, test it on the remaining fraction of training data and average the error

bootstrapping

extracting of observations with replacement. out of this dataset you can extract samples, put them back, same observation can be extracted multiple times. Phillip will touch on this later.

Hide

```
include_graphics("Bootstrapping.png")
```

Bootstrapping [2]

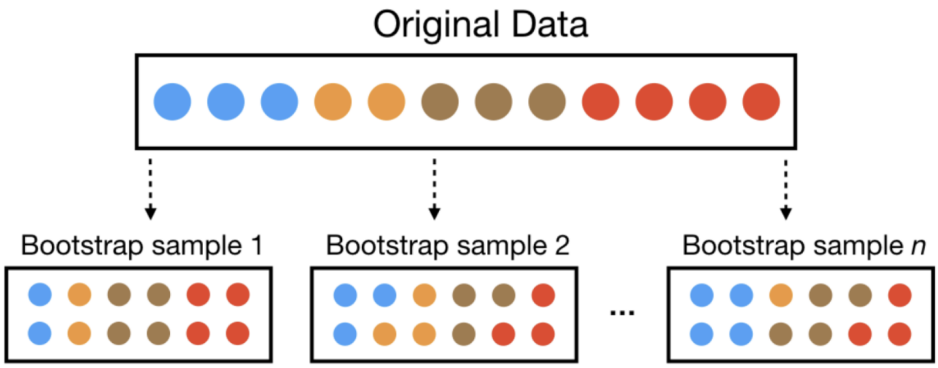


Figure 2.6: Illustration of the bootstrapping process.

[2] Iteratively train each model (f) on a sample taken *with replacement* of the original data and average the error

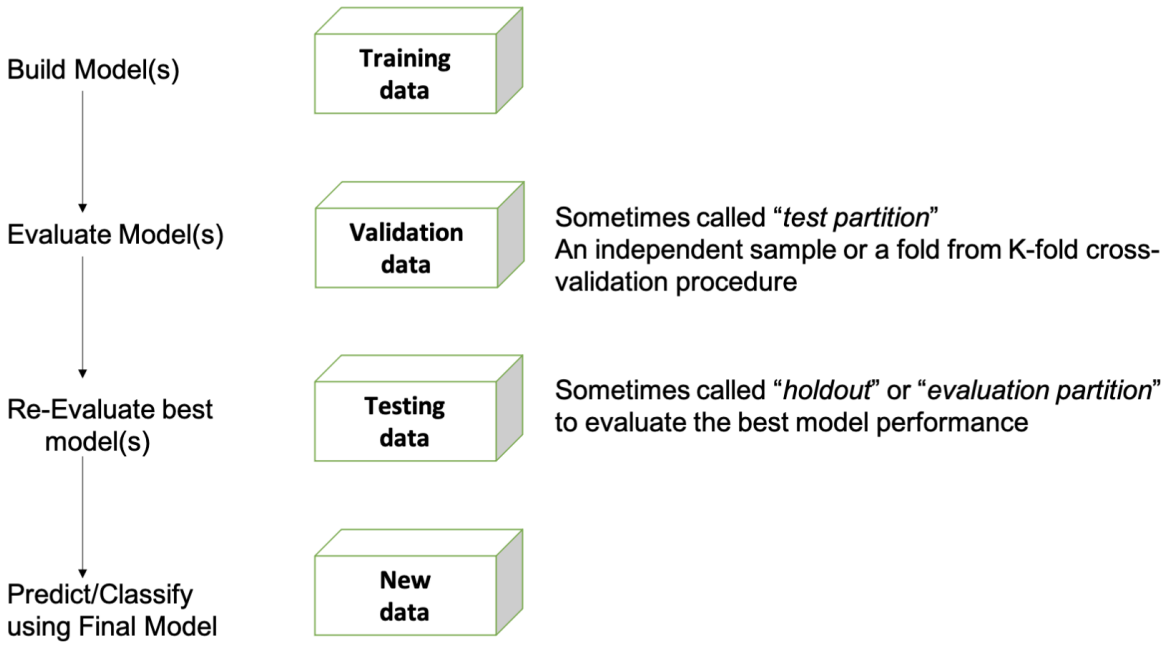
Use multiple split when practicing to get best results

Knowing this, the standard procedure for model building should be

Hide

```
include_graphics("Procedure modelling.png")
```

Summarizing



Source: Adapted from Shmueli, et al., 2018, Data Mining for Business Analytics

Model evaluation criteria

###will be expanded upon

expand with knowledge from https://bradleyboehmke.github.io/HOML/process.html (https://bradleyboehmke.github.io/HOML/process.html)

Regression models:

bold = most common

- **MSE** (mean squared error)
- **RMSE** (root mean squared error)
- Deviance
- MAE
- R-squared

Classification models

- Misclassification rate
- Mean per class error
- MSE
- Cross-entropy
- Gini index
- Confusion matric
- Accuracy, Precision, Sensitivity/Recall, Specificity
- ROC and AUC*

Errors

Training errors(MSE)

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

Training error(RMSE)

$$\sqrt{MSE}$$

Testing error (test MSE)

$$Ave(y_o - \hat{f}(x_o))^2$$

where x_o, y_o are obs. not used to train. Otherwise formulas are pretty much the same

overfitting

Small train error

High test error

Our model(algorithm) is trying too hard to find a suited fit

Variance of fit = Amount by which $\hat{f}$ would change if estimated using different training set

bias of fit = error introduced by approximating real-life problem in simple models

aim to minimize both

$$E(y_0 - \hat{f}(x_o))^2 = Var(\hat{f}(x_o)) + [bias(\hat{f}(x_o))]^2 + Var(e)$$

where

$E(y_0 - \hat{f}(x_o))$ = Expected test MSE

$Var(\hat{f}(x_o$ = Variance of fit

$[bias(\hat{f}(x_o))]$ = Bias of fit

$Var(e)$ = Irreducible error

Hide

```
include_graphics("Method choice example.png")
```

Example

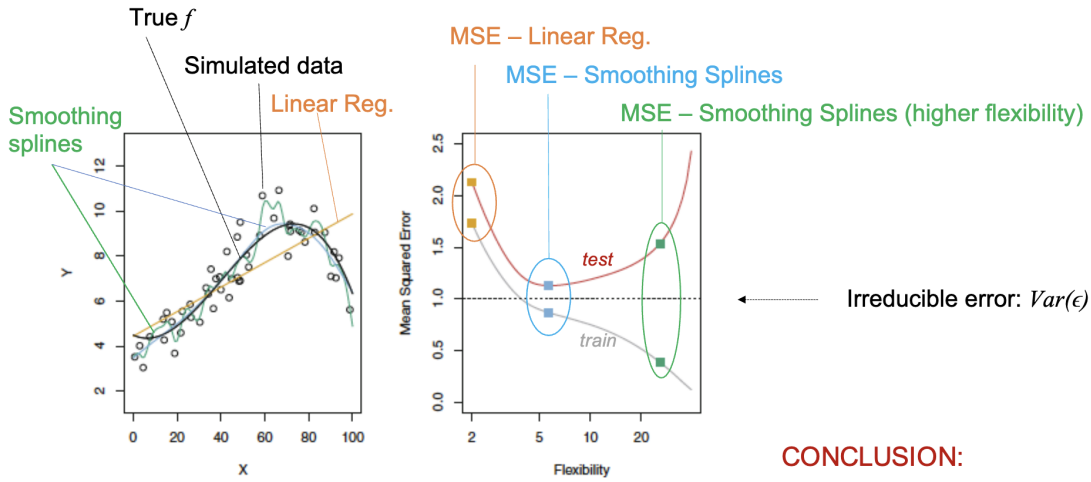


FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

CONCLUSION:

As flexibility (model complexity) increases,

- training MSE decreases
- test MSE decreases but increases again – notice the *U-shape.*

On the example we have three models fitted. Dataset was simulated based on black (true f).Smoothing spines on left graph is too aggressively trying to fit datapoints.

For the linear regression the training error is the lower yellow square, testing errorr upper yellow square. These are the errors when we run a LM on the dataset. For blue and green we see same relationship. As shown here a higher flexibility model is not laways the optimal choice. **In the example we would go for the blue MSE to avoid overfitting**.

We always aim for lowest test error.

Lecture conclusions

Increase in method flexibility (more advanced methods, NN), we can reduce the prediction error (bias). Increasing flexibility does however have diminishing returns and will eventually increase our variance further than reducing our bias.

Machine learning has to one-size-fits-all model, we must utilize all tools and models available to us to treat each dataset independently.

Lecture 4 coding 🤓

Sample formula interfaces

Hide

```
ames <- AmesHousing::make_ames()
# Sale price as function of neighborhood and year old

lm_lm <- lm(Sale_Price ~Neighborhood + Year_Sold, data = ames)

#lm is used to fit linear models

lm_glm <- glm(Sale_Price ~Neighborhood + Year_Sold, data = ames, family = gaussian)

#glm is used to fit generalized linear models

lm_caret <- train(Sale_Price ~Neighborhood + Year_Sold, data = ames, method = "lm")
```

```
Warning: prediction from a rank-deficient fit may be misleadingWarning: prediction from a ran
k-deficient fit may be misleadingWarning: prediction from a rank-deficient fit may be mislead
ingWarning: prediction from a rank-deficient fit may be misleadingWarning: prediction from a
rank-deficient fit may be misleadingWarning: prediction from a rank-deficient fit may be misl
eadingWarning: prediction from a rank-deficient fit may be misleadingWarning: prediction from
a rank-deficient fit may be misleadingWarning: prediction from a rank-deficient fit may be mi
sleadingWarning: prediction from a rank-deficient fit may be misleadingWarning: prediction fr
om a rank-deficient fit may be misleadingWarning: prediction from a rank-deficient fit may be
misleadingWarning: prediction from a rank-deficient fit may be misleadingWarning: prediction
from a rank-deficient fit may be misleadingWarning: prediction from a rank-deficient fit may
be misleadingWarning: prediction from a rank-deficient fit may be misleadingWarning: predicti
on from a rank-deficient fit may be misleadingWarning: prediction from a rank-deficient fit m
ay be misleadingWarning: prediction from a rank-deficient fit may be misleadingWarning: predi
ction from a rank-deficient fit may be misleadingWarning: prediction from a rank-deficient fi
t may be misleadingWarning: prediction from a rank-deficient fit may be misleadingWarning: pr
ediction from a rank-deficient fit may be misleadingWarning: prediction from a rank-deficient
fit may be misleadingWarning: prediction from a rank-deficient fit may be misleading
```

Hide

```
lm_caret
```

```
Linear Regression

2930 samples
   2 predictor

No pre-processing
Resampling: Bootstrapped (25 reps)
Summary of sample sizes: 2930, 2930, 2930, 2930, 2930, 2930, ...
Resampling results:

  RMSE      Rsquared   MAE
  52606.52  0.5646596  35649.72

Tuning parameter 'intercept' was held constant at a value of TRUE
```

Hide

```
#train used as part of the Caret library. Documentation found in tfestimators package
```

In class with Ana follow-along

Tutorial 2

Problem 1: Programming The purpose of this problem is to get comfortable with R and its facilities. We shall spendmost of the time doing some basic computations. If you are a good programmer you will finishthese computations quickly. First start by opening R, create a new script and save it to your hard drive with the name: ”Exercise1.R”.

Part 1

Hide

```
v1 <- c(1,2,2,1)
v2 <- c(2,3,3,2)

v1+v2
```

```
[1] 3 5 5 3
```

Hide

```
v1-v2
```

```
[1] -1 -1 -1 -1
```

Hide

```
v1*v2
```

```
[1] 2 6 6 2
```

Hide

```
v3 <- c(v1,v2)
```

Part 2

Hide

```
#1
m1 <- c(1,6,3,2,4,6)
mA <- matrix(m1,ncol=2)
mA
```

```
     [,1] [,2]
[1,]    1    2
[2,]    6    4
[3,]    3    6
```

Hide

```
#2
print(mA[1,])
```

```
[1] 1 2
```

Hide

```
print(mA[2,])
```

```
[1] 6 4
```

Hide

```
print(mA[3,])
```

```
[1] 3 6
```

Hide

```
print(mA[,1])
```

```
[1] 1 6 3
```

Hide

```
print(mA[,2])
```

```
[1] 2 4 6
```

Hide

```
rowSums(mA)
```

```
[1]  3 10  9
```

Hide

```
apply(mA, 1, FUN=min)
```

```
[1] 1 4 3
```

Hide

```
apply(mA, 1, FUN=max)
```

```
[1] 2 6 6
```

Hide

```
sort(mA[,1],decreasing = FALSE)
```

```
[1] 1 3 6
```

Hide

```
#3

mD <- matrix(1:1, ncol= 4, nrow=4)

mD[c(1,6,11,16)] <- 0
mD
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    1    1    1
[2,]    1    0    1    1
[3,]    1    1    0    1
[4,]    1    1    1    0
```

Hide

```
mD <- matrix(1, nrow=4, ncol=4)

diag(mD) <- rep(0, nrow(mD))
mD
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    1    1    1
[2,]    1    0    1    1
[3,]    1    1    0    1
[4,]    1    1    1    0
```

Hide

```
mE <- matrix(1:16, ncol=4,nrow=4,byrow=TRUE)
mE
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    2    3    4
[2,]    5    6    7    8
[3,]    9   10   11   12
[4,]   13   14   15   16
```

Hide

```
mE[-c(3,5,6,9,16)] <- 0

mE
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    2    3    0
[2,]    0    6    0    0
[3,]    9    0    0    0
[4,]    0    0    0   16
```

Hide

```
mi <- diag(x=1, nrow=4, ncol=4)


#4

mF <- (rbind(mD,mE))
mF
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    1    1    1
[2,]    1    0    1    1
[3,]    1    1    0    1
[4,]    1    1    1    0
[5,]    0    2    3    0
[6,]    0    6    0    0
[7,]    9    0    0    0
[8,]    0    0    0   16
```

Hide

```
mE+mD
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    3    4    1
[2,]    1    6    1    1
[3,]   10    1    0    1
[4,]    1    1    1   16
```

Hide

```
mE*mD
```

```
     [,1] [,2] [,3] [,4]
[1,]    0    2    3    0
[2,]    0    0    0    0
[3,]    9    0    0    0
[4,]    0    0    0    0
```

Hide

```
mE %*% mD #matrix product
```

```
     [,1] [,2] [,3] [,4]
[1,]    5    3    2    5
[2,]    6    0    6    6
[3,]    0    9    9    9
[4,]   16   16   16    0
```

Hide

```
#5
x = 1
calc_x <- {
    if(x <= 0) {
      print("-x^3")
    } else {
      if(x > 1) {
        print("sqrtx")
       } else {print("x^2")
       }
    }
}
```

```
[1] "x^2"
```

Hide

```
calc_x
```

```
[1] "x^2"
```

Hide

```
#6 busted måde
# h(x,n)=1 +x+x2+x3+···+xn=∑ni=0xi
# using replicate it's easy to match x to n

#func <- function(hxn)
 # {
#for (j in 1:n)
#{
#  x[j] = j^n
#}
# x
#}

#n = 6
#x_1 = n
#x = rep(x_1,n)

#func(hxn)

#6
func <- function(x,n)
{
  sum = 0

  for (j in 0:n)
  {
    sum = sum + x^j
  }
  return(sum)
}

func(x=3, n=3)
```

```
[1] 40
```

Hide

```
# 2^0 + 2^1 + 2^2 + 2^3

#7 fuck while loops

func2 <- function(x,n)
{
  sum = 0
  j = 0
  while (j <= n)
  {
    sum = sum + x^j
    j = j + 1
  }
  return(sum)
}

func2(x=3, n=3)
```

```
[1] 40
```

Hide

```
func3 <- function(x,n)
{
x1 <- c(0:x)
print(x1)
{
  n1 <- (0:n)
  print(n1)
}
nx1 <- x1^n1
nx1
print(sum(nx1))
}

func3(x=3, n = 3)
```

```
[1] 0 1 2 3
[1] 0 1 2 3
[1] 33
```

Hide

```
  #8

# A room contains 100 toggle switches, originally all turned off.  100 people enter the roomi
n turn.  The first one toggles every switch, the second one toggles every second switch, theo
ne third every third switch, and so on until the last person, who toggles the last switch onl
y.  At the end of this process, which switches are turned on?Note:This requires alittle think
ing. Don’t give up!

#rest state = 100 off
#first pass = 100 on
#second pass = 50 on, c(1:100,2) is on
#third pass =


2nd pass = seq(1,100,by=2)
```

```
Error: unexpected symbol in "2nd"
```