Instead of recordeing the foundamental knowledge in DSA1101, this cheatsheet is focusing on the real part in the code implementation or some "how to interpret" questions...
Mainly stems from the lecture slides, Tutorial slides, on-site, off-site questions and so on.

R introduction

Elements in vectors and matrices must be the same mode
Dataframe is similar to the Matrix object but columns can have different modes.

Where do we use list topic 1 page 9

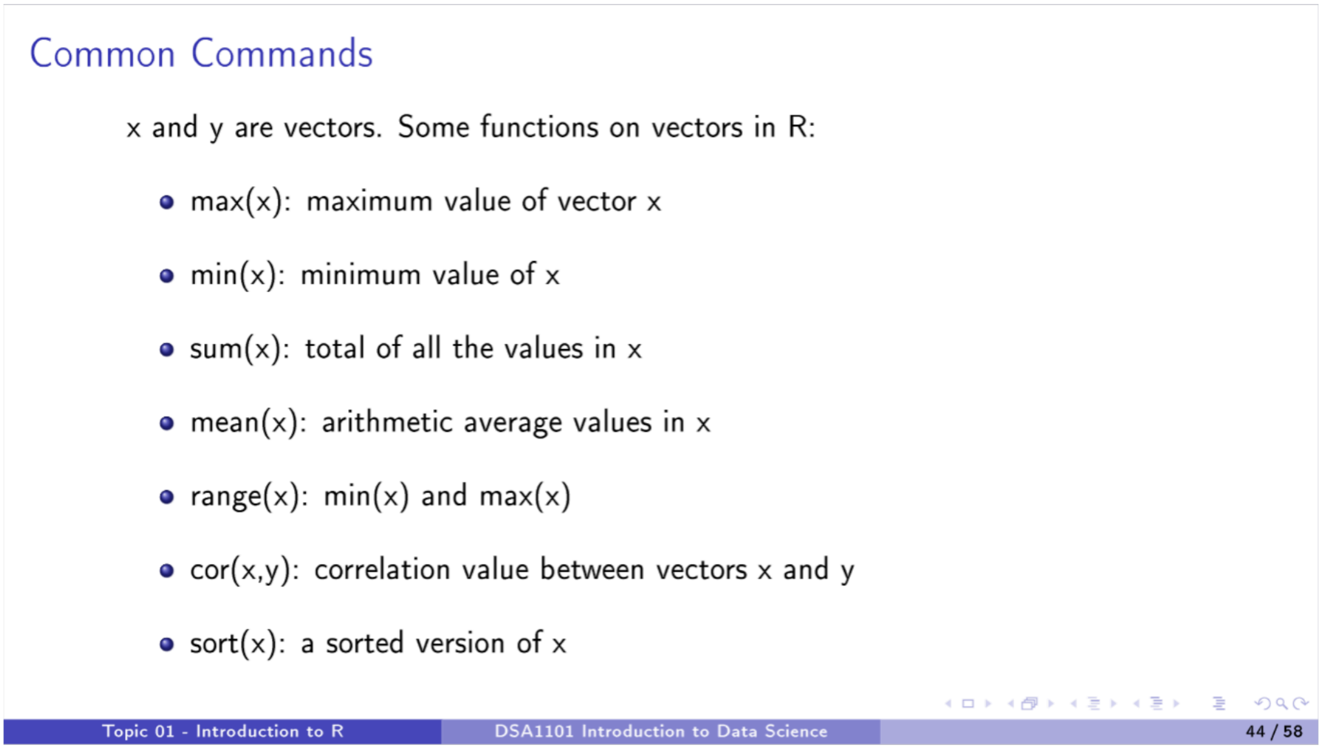
```
df[1:4,2:3]
# use square bracket to index dataframe

# more general, assessing parts of a df by conditions
df[gender == 'male',] # don't forget the ','

# if we want more conditions
df[gender == 'male' & CA2 > 85,]
```

read files: topic 1 page 30

When should we use attach(df) topic page 38

Common commands

`nrow(df)` report the num of rows
`ncol(df)` report the num of cols



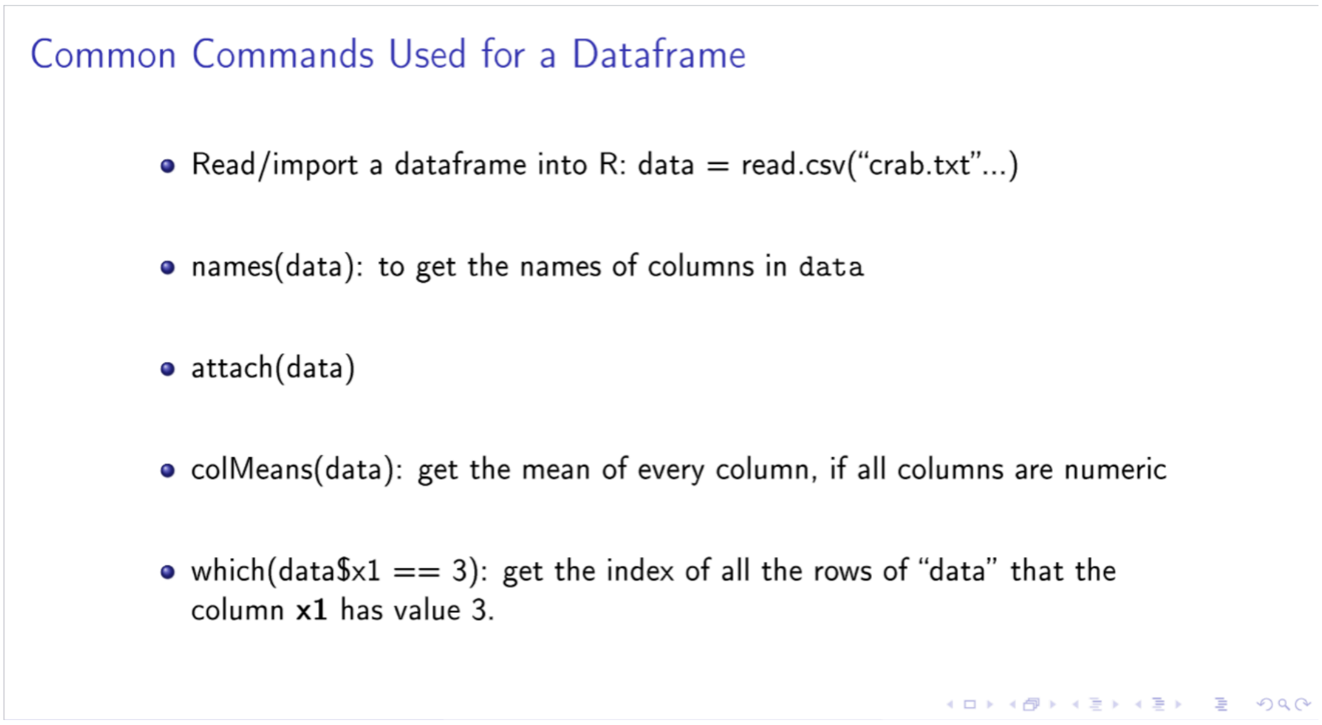
Common Commands

x and y are vectors. Some functions on vectors in R:

- max(x): maximum value of vector x
- min(x): minimum value of x
- sum(x): total of all the values in x
- mean(x): arithmetic average values in x
- range(x): min(x) and max(x)
- cor(x,y): correlation value between vectors x and y
- sort(x): a sorted version of x

Topic 01 - Introduction to R DSA1101 Introduction to Data Science 44 / 58

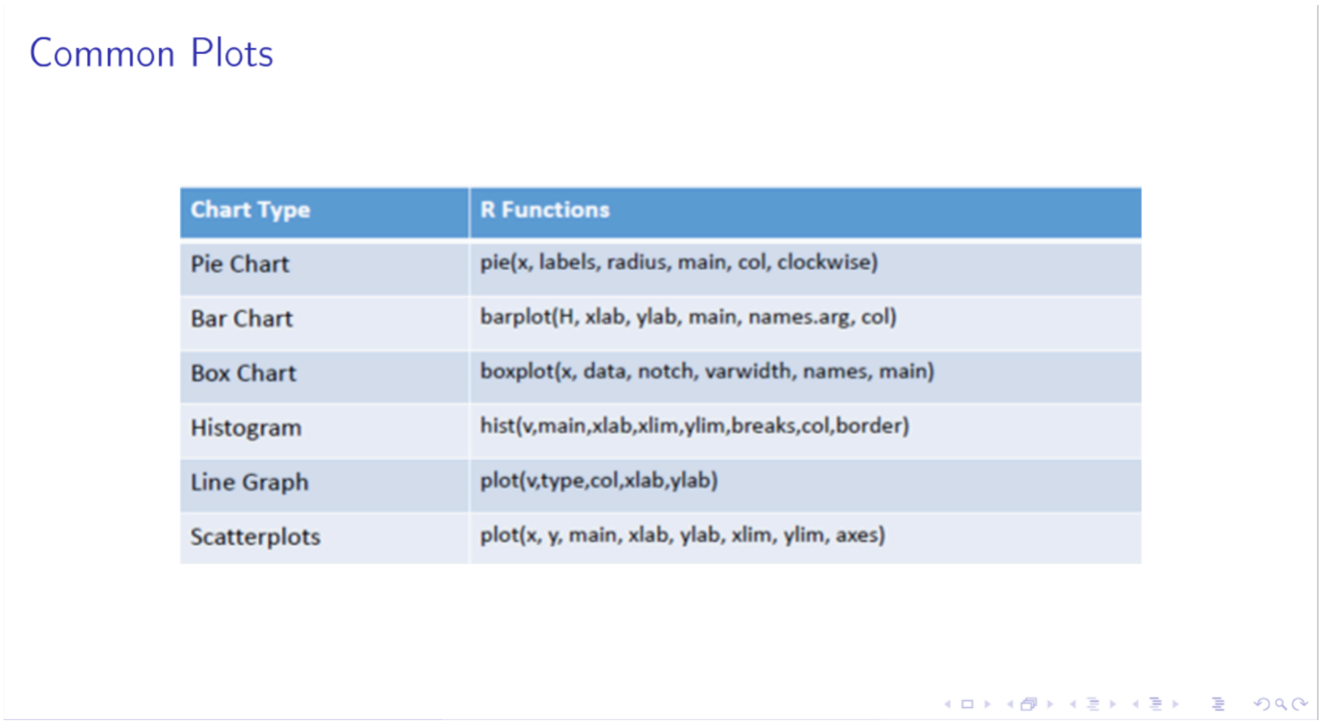
use of correlation value?



Common Commands Used for a Dataframe

- Read/import a dataframe into R: data = read.csv("crab.txt"...)
- names(data): to get the names of columns in data
- attach(data)
- colMeans(data): get the mean of every column, if all columns are numeric
- which(data$x1 == 3): get the index of all the rows of "data" that the column **x1** has value 3.

which gives us the index



Common Plots

Chart Type	R Functions
Pie Chart	pie(x, labels, radius, main, col, clockwise)
Bar Chart	barplot(H, xlab, ylab, main, names.arg, col)
Box Chart	boxplot(x, data, notch, varwidth, names, main)
Histogram	hist(v,main,xlab,xlim,ylim,breaks,col,border)
Line Graph	plot(v,type,col,xlab,ylab)
Scatterplots	plot(x, y, main, xlab, ylab, xlim, ylim, axes)

Create a box chart

Q: Create box plots of car's price by groups of year for year 2016 to 2020. **Give your comments**

?what should be in the comments?

```
boxplot(price ~ year, # x: 公式 (Y ~ Group)
        data = df[df$year >= 2016 & df$year <= 2020,] # 数据源
        main = "Car Prices by Year", # 标题 (必填)
        xlab = "Year",
        ylab = "Price",
        # --- 进阶参数 ---
        notch = TRUE, # 开启"收腰": 缺口不重叠 = 中位数显著不同
        varwidth = TRUE, # 开启"变宽": 箱子越胖 = 数据量(n)越大
        col = "lightblue"
        )
```

Comments:

- **Trend**. e.g: The median price seems to increase from 2016 to 2020. (the black line of the box)
- **Spread**. e.g: The variation in prices is larger in 2020 compared to 2016. (the height of the box IQR)
- **Outliers**. e.g: There are significant outliers on the higher end for all years.

Loops, conditions and functions

Loops

- while loop

```
while (condition) {expression}
```

Data wrangling

How to change a lot of values in one column

~~SOP:

- ~~get access to the specific column~~
- ~~which filter the specific index of what we want
- ~~how can we change those value? for loop?

we should avoid use **for loop** in R, 'cause R is a vectorized language, we could do batch manipulation

and the original SOP is a bit complex

```
# structure: df$column[condition] = new value
df$column[df$column == "Other"] = "Hybrid"
```

Remove data out of our scope (not remove but keep)

SOP:

- clarify the conditions

```
# we don't execute remove, but execute "keep/filter"
data = data[data$year <= 2026, ]
```

Create Contingency Table and report number in specific condition

~~SOP~~

- ~~clarify the conditions
  ~~
  Instead, we need to use correct function `table()`

```
tab1 = table(row_variable, col_variable) # cols in df are vectors
tab1 = table(data$transmission, data$fuelType)

print(tab1) # use this line to check

# report specific number
tab1["RowName", "ColName"]
tab1["Automatic", "Petrol"]

addmargins(tab1) # 给表加上 Total 行和 Total 列
```

Calculate probability in table

core func `prob.table(table_object, margin)`

- margin = 1, let the sum of every row = 1
- margin = 2, let the sum of every col = 1

```
prob_tab = prop.table(tab2, margin = 2)
print(prob_tab)

# we could use colSums to check
```

```
# 提取概率
p_auto_high <- prob_tab["high", "Automatic"]
p_manual_high <- prob_tab["high", "Manual"]
# 计算差值
diff <- p_auto_high - p_manual_high
print(diff)
```

How to interpret the difference?

- **The probability of a car being high-priced is 0.45 higher for automatic cars compared to manual cars.**

Create new column by specific condition

```
# df&col = ifelse(条件, 条件成立时的值, 条件不成立时的值)
data$priceHL = ifelse(data$price > 12500, "high", "low")
```

General Utilities and Answer Formatting

```
format_ans = function(x){
    if (abs(x) < 1){
        return (signif(x,3))
    } else {
        return (format(round(x,3), nsmall = 3))
    }
}
```

Linear Regression

SLR(Simple Linear Regression)

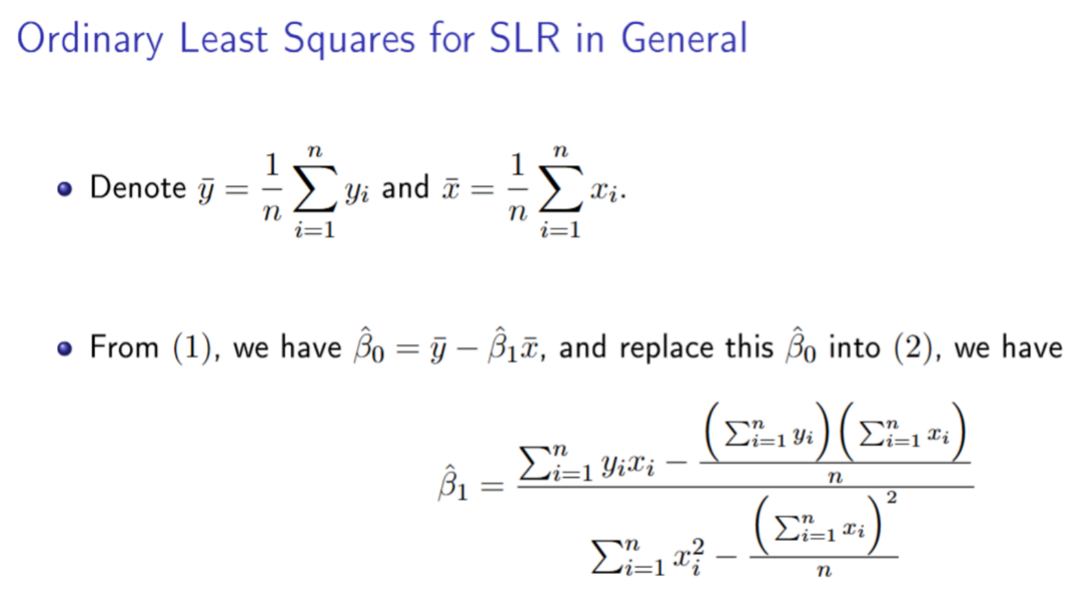
To express the total magnitude of the deviations, we sum up the squared residuals for all the data points, **Residual Sum of Squares**, abbreviated as **RSS**

- Consider RSS as a function in terms of β_0 and β_1. Let's call it $h(\beta_0, \beta_1)$.

- To find the minimum of $h(\beta_0, \beta_1)$, we first take the derivative of it w.r.t β_0 while holding β_1 constant, and then take the derivative of it w.r.t β_1 while holding β_0 constant.

$$\frac{\partial h(\beta_0, \beta_1)}{\partial \beta_0} = -11 + 6\beta_0 + 14\beta_1.$$
$$\frac{\partial h(\beta_0, \beta_1)}{\partial \beta_1} = -53 + 14\beta_0 + 70\beta_1.$$

and set the partial derivatives to zero
The whole process(to get and) is known as the method of **Ordinary Least Squares** (OLS)



Ordinary Least Squares for SLR in General

- Denote $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ and $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.
- From (1), we have $\hat{\beta}_0 = \bar{y} - \hat{\beta}_1\bar{x}$, and replace this $\hat{\beta}_0$ into (2), we have

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} y_i x_i - \frac{\left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i\right)}{n}}{\sum_{i=1}^{n} x_i^2 - \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n}}$$

After the math background, we are heading to build the model, and use it to predict

```
# build
M = lm(vec1 ~ vec2, data = df)

>>> return intercept x(slope)

# create a new dataframe
new_df = data.frame(
    vec1 = c(),
    vec2 = c()
) # names in the new_df must be the same as those in the model
# if we are predicting only one data point, we don't need to use c(), but it
doesn't matter, lol

# predict
predict(model_name, newdata = new_df)
```

What if...

```
# use of I()

#
```

Goodness-of-fit of the model

run:

```
summary(model_name)
```

to see the result

F-test

To test if the whole model is significant or not

null hypothesis(): all the coefficients, except intercept, are zero

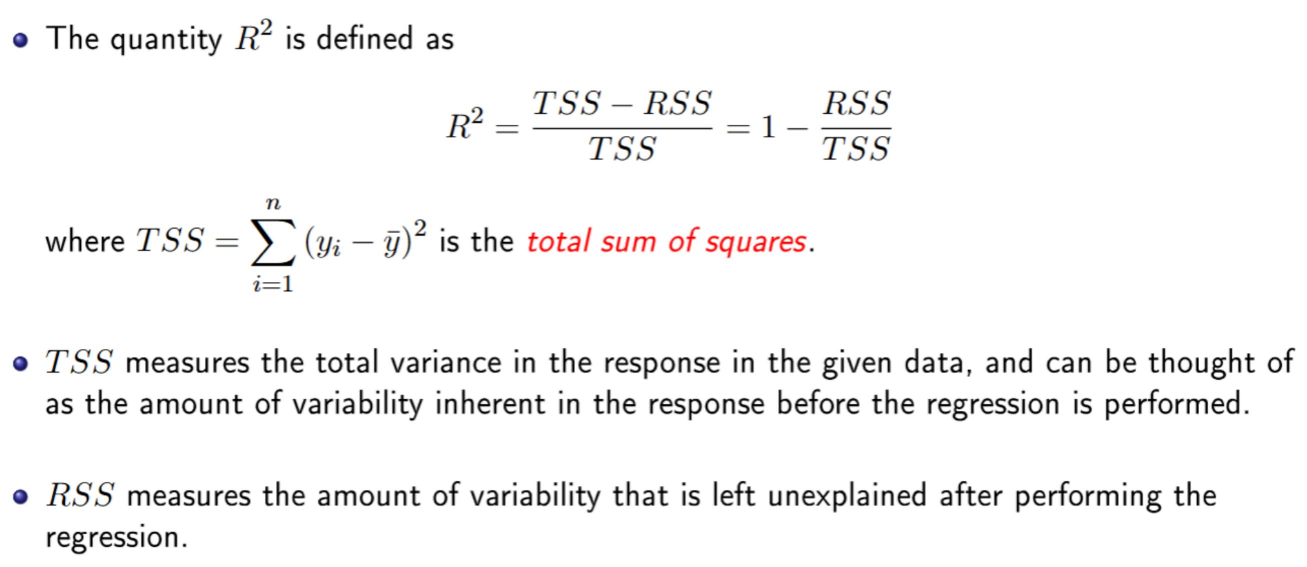
If the test has small p-value (such as <0.05), then data provide strong evidence against
. Otherwise, we cannot eliminate

```
F-statistic: 4.631 on 1 and 1 DF, p-value: 0.2769
```

- 4.361: Signal/Noise
- 1(first): Model DF, the parameter used in the model
- 1(second): Residual DF, (总样本数 - 变量数 - 1)

注意样本量太少的情况，结果可能不可靠

Coefficient of determination



- The quantity R^2 is defined as

$$R^2 = \frac{TSS - RSS}{TSS} = 1 - \frac{RSS}{TSS}$$

where $TSS = \sum_{i=1}^{n} (y_i - \bar{y})^2$ is the *total sum of squares*.

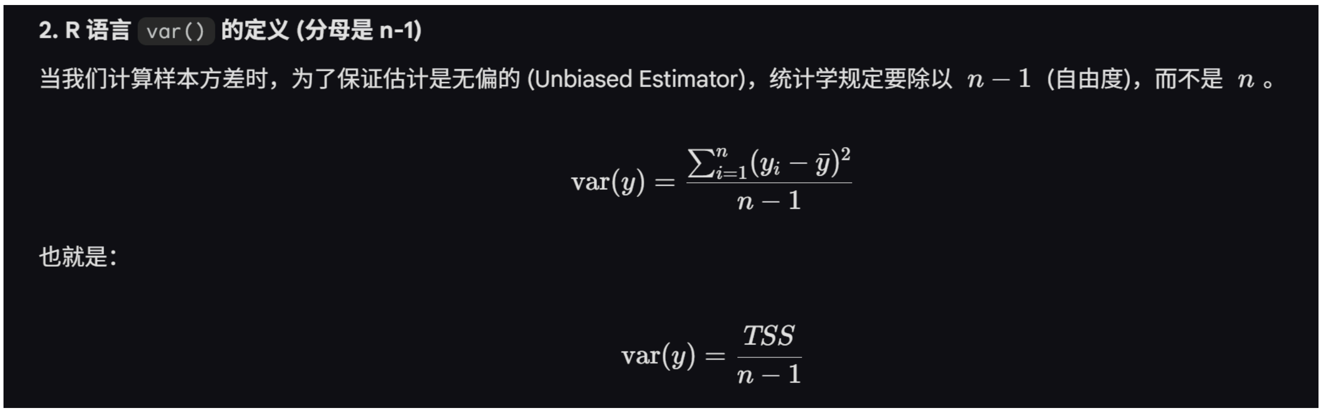
- TSS measures the total variance in the response in the given data, and can be thought of as the amount of variability inherent in the response before the regression is performed.

- RSS measures the amount of variability that is left unexplained after performing the regression.

= 模型成功解释的比例
close to 1 means the model fitted well

use formula to solve R-squared

```
> RSS =sum ((y- M$fitted )^2)
> TSS = var(y)*(length(y) -1)
> R2 = 1 - RSS/TSS;
```



2. R 语言 `var()` 的定义 (分母是 n-1)

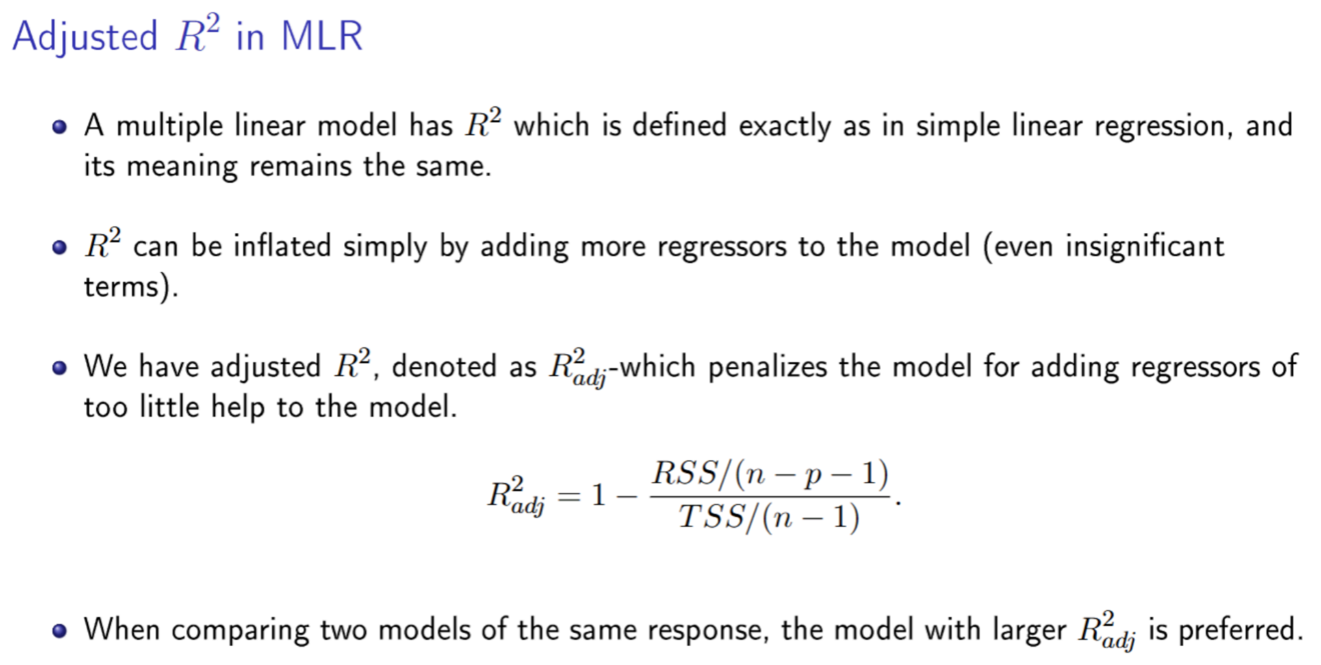
当我们计算样本方差时，为了保证估计是无偏的 (Unbiased Estimator)，统计学规定要除以 $n-1$ (自由度)，而不是 n 。

$$\mathrm{var}(y) = \frac{\sum_{i=1}^{n}(y_i - \bar{y})^2}{n-1}$$

也就是：

$$\mathrm{var}(y) = \frac{TSS}{n-1}$$

When we only have 1 variable, check **Multiple R-squared**, otherwise, check **adjusted R-squared**(which has the penalty mechanism)



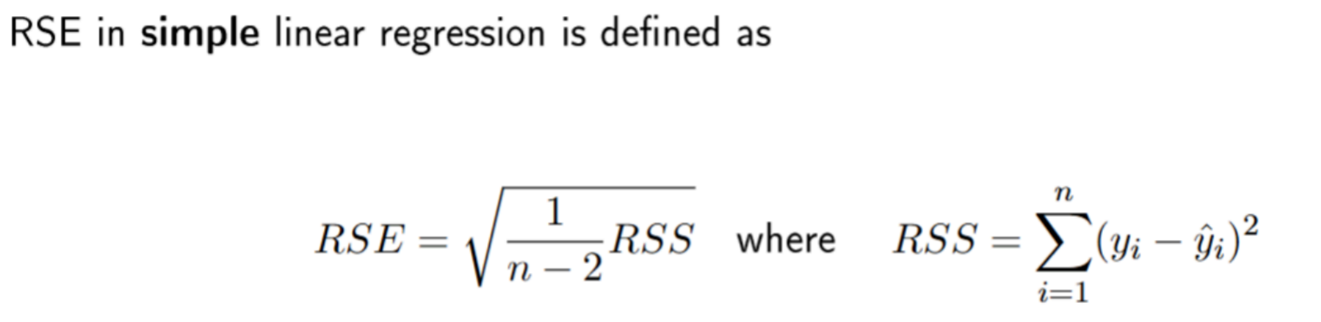
Adjusted R^2 in MLR

- A multiple linear model has R^2 which is defined exactly as in simple linear regression, and its meaning remains the same.
- R^2 can be inflated simply by adding more regressors to the model (even insignificant terms).
- We have adjusted R^2, denoted as R^2_{adj}-which penalizes the model for adding regressors of too little help to the model.

$$R^2_{adj} = 1 - \frac{RSS/(n-p-1)}{TSS/(n-1)}.$$

- When comparing two models of the same response, the model with larger R^2_{adj} is preferred.

Residual Standard Error (RSE)



RSE in **simple** linear regression is defined as

$$RSE = \sqrt{\frac{1}{n-2}RSS} \quad \text{where} \quad RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Multiple Linear Regression

```
# build the model
m = lm(y~x1 + x2, data = df)
```

Find the correlation between variable1 and variable2

`cor(x,y)`

```
# 1. Calculate correlation between Price and Mileage
cor_1 = cor(1/data$price, data$mileage)
print(cor_1)
```

?如果数据里有 `NA`，记得加上 `use = "complete.obs"`？

What to do when having categorical variable

无脑加 `as.factor()`

```
model = lm(price ~ as.factor(fuelType) + as.factor(transmission) + mileage,
data = df)
```

KNN

Fundamental Algorithm

just memorize the data, without training

- The k-nearest neighbors method uses training points closest in feature space to x to find

$$\frac{1}{k}\sum_{i\in\mathcal{N}_k(x)} y_i,$$

where $\mathcal{N}_k(x)$ is the neighborhood of x defined as the set of k closest points (in terms of Euclidean distance).

- δ is a pre-chosen threshold, the average above is converted to $\widehat{G}(x)$ according to the rule

$$\widehat{G}(x) = \begin{cases} 1, & \text{if it's} > \delta, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Bias-variance tradeoff(what will happen when k changes)

- When $k = 1$, the decision boundary is overly flexible and influenced by local features of a handful of training data points $\rightarrow$ low bias, high variance.

- When $k = 100$, the method yields more stable but less flexible decision boundaries $\rightarrow$ high bias, low variance.

Bias

Distance from the model to the function we want to fit
High means that the model's assumption is too strong(such as linear model, which assumes that the world is linear)

Variance

Model over-relies on the specific data we used

单个模型 偏离它自己平均水平 的程度

Application

use the `knn()` function from the `class` package in R

Inputs

- A **matrix** containing the predictors or features associated with the training data
- A **matrix** containing the predictors or features associated with the data for which we wish to make predictions
- A **vector** containing the class labels for the training data
- A **value** for , the number of nearest neighbors to be used by the classiffier

Data Preprocessing

- extract particular cols
- standarization

```
# 1. 挑选出需要标准化的列 (Numeric features)
cols_to_scale = c("mileage", "tax", "mpg", "engineSize")

# 2. 创建新 data.frame
# 注意顺序: 题目要求第一列是 priceHL
data.KNN = data.frame(
  priceHL = data$priceHL,           # 第一列: 响应变量 (不变)
  scale(data[, cols_to_scale])      # 后几列: 标准化后的特征
)

# 3. (可选但推荐) 检查一下结果
head(data.KNN)
```

Train-test split

```
# 1. 锁定随机种子 (必须先运行这行, 否则每次抽样结果都不一样)
set.seed(210)
```

```
# 2. 生成随机索引 (Random Indices)
# 逻辑: 从 1 到 总行数 中, 随机抽取 一半(50%) 的数字
n_rows = nrow(data.KNN)
train.index = sample(1:n_rows, n_rows/2) # 1st parameter: range, 2nd
parameter: number of the index

# 3. 创建 Train Set (使用正索引)
train.set = data.KNN[train.index, ]

# 4. 创建 Test Set (使用负索引 - 这里的减号 "-" 意思是 "exclude/排除")
test.set = data.KNN[-train.index, ]
```

In practice

```
# 0. 记得先加载包 (KNN 需要 class 包)
library(class)

# 1. 创建 K 的向量 (3, 5, ..., 25)
K <- seq(3, 25, 2)

# 2. 初始化存放结果的空向量 (Pre-allocate memory)
# 这一步不是必须的, 但是是"好习惯", 能让代码跑得更快
fnr <- numeric(length(K))
accuracy <- numeric(length(K))

# 3. 锁定随机种子 (题目要求)
set.seed(210)

# 4. 开始循环
for(i in 1:length(K)) {

  # 当前的 k 值
  k_curr <- K[i]

  # --- 运行 KNN ---
  # train: 训练集特征 (排除第一列 Response)
  # test:  测试集特征 (排除第一列 Response)
  # cl:    训练集的标签 (第一列 Response)
  knn.pred <- knn(train = train.set[, -1],
                  test = test.set[, -1],
                  cl = train.set[, 1],
                  k = k_curr)

  # --- 建立混淆矩阵 (Confusion Matrix) ---
  # table(真实值, 预测值)
  cm <- table(test.set[, 1], knn.pred)

  # --- 计算 Accuracy ---
  # (TP + TN) / Total
```

```r
  # sum(diag(cm)) 是对角线之和 (预测对的数量)
  accuracy[i] <- sum(diag(cm)) / sum(cm)

  # --- 计算 FNR (False Negative Rate) ---
  # 题目定义: High-price is Positive.
  # FNR = FN / (TP + FN) = 漏报 / 所有真的High

  # 假设矩阵是 cm["high", "high"] 这种格式
  # FN (False Negative): 实际上是 "high", 但预测成了 "low"
  # 注意: 如果你的 levels 顺序不同, 这里的索引位置可能不同, 最好用名字索引
  FN <- cm["high", "low"]
  TP <- cm["high", "high"]

  fnr[i] <- FN / (TP + FN)
}

# 5. 查看结果
# print(data.frame(K, accuracy, fnr))
```

Further processing

Magic of Logical Indexing

```r
# --- 第一部分: 找出 FNR < 0.1 的所有 K ---

# 1. 创建逻辑索引 (Logical Indexing)
# 这一步会生成一个 TRUE/FALSE 的向量, 告诉我们哪个位置满足条件
condition <- fnr < 0.1

# 2. 提取满足条件的数据, 组装成矩阵
# cbind (Column Bind) 把三列拼在一起
good.fnr <- cbind(K = K[condition],
                  fnr = fnr[condition],
                  accuracy = accuracy[condition])

# 3. 查看并报告结果 (按题目要求写在注释里)
# print(good.fnr)


# --- 第二部分: 在好学生里选 Accuracy 最高的 ---

# 1. 在 good.fnr 矩阵里, 找到 accuracy 那一列最大值的"位置" (Index)
# good.fnr[, "accuracy"] 取出准确率这一列
# which.max() 返回最大值所在的行号
best_idx <- which.max(good.fnr[, "accuracy"]) # importantv

# 2. 提取对应的 K 值
# 取出 best_idx 这一行的 "K" 列
best.K <- good.fnr[best_idx, "K"]
```

```r
# 3. 报告 best.K 的详细信息
# print(good.fnr[best_idx, ])
# 或者手动写: best_stats <- good.fnr[best_idx, ]
```

Predict single data point

```r
# 1. 准备原始数据统计量 (Original Statistics)
# 注意: 一定要对应好列的顺序! (mileage, tax, mpg, engineSize)
cols <- c("mileage", "tax", "mpg", "engineSize")

# 计算原始数据的均值和标准差
mu <- colMeans(data[, cols])  # 这里的 data 是你最原始的 data frame
sigma <- apply(data[, cols], 2, sd)

# 2. 准备新数据点 (New Observation)
# 先把它写成一个跟原来一样的 data frame (未缩放的原始值)
new_car_raw <- data.frame(
  mileage = 30000,
  tax = 110,
  mpg = 60,
  engineSize = 1
)

# 3. 手动标准化 (Manual Standardization)
# 公式: (New - Mean) / SD
new_car_scaled <- data.frame(
  mileage    = (30000 - mu["mileage"]) / sigma["mileage"],
  tax        = (110   - mu["tax"])     / sigma["tax"],
  mpg        = (60    - mu["mpg"])     / sigma["mpg"],
  engineSize = (1     - mu["engineSize"]) / sigma["engineSize"]
)

# 另一种更骚气的写法 (向量化操作, 考试如果想省时间可以用):
# new_car_scaled <- new_car_raw
# for(col in cols) {
#    new_car_scaled[col] <- (new_car_raw[col] - mean(data[,col])) /
sd(data[,col])
# }

# 4. 进行预测 (KNN Prediction)
# 这里的 train 和 cl 还是用原来的
# test 换成我们刚算出来的 new_car_scaled
library(class)
pred_label <- knn(train = train.set[, -1],  # 训练集特征
                  test = new_car_scaled,    # 测试集 (我们的新车)
                  cl = train.set[, 1],      # 训练集标签
                  k = best.K)               # 用你上一题算出来的最佳 K
```

```
# 5. 报告结果
print(pred_label)
```

In real world

```
# 正确的做法 (Real World)
# 1. 先切分
train_idx <- sample(1:nrow(data), nrow(data)/2)
train_raw <- data[train_idx, ]
test_raw  <- data[-train_idx, ]

# 2. 只从 Train 算统计量
train_mean <- mean(train_raw$mileage)
train_sd   <- sd(train_raw$mileage)

# 3. 应用到 Train
train_scaled <- (train_raw$mileage - train_mean) / train_sd

# 4. 应用到 Test (注意: 这里用的必须是 train_mean, 不能用 test_mean!)
test_scaled  <- (test_raw$mileage  - train_mean) / train_sd
```

Diagnostics

- In general, for two class labels, C and C', where C' denotes "not C", some working definitions and formulas follow:

- True Positive: Predict C, when actually C
- True Negative: Predict C', when actually C'
- False Positive: Predict C, when actually C'
- False Negative: Predict C', when actually C

Confusion Matrix

- In a two-class classification, confusion matrix is constructed as below.

		Predicted Class	
		Positive	**Negative**
Actual Class	**Positive**	True Positives (TP)	False Negatives (FN)
	Negative	False Positives (FP)	True Negatives (TN)

5 Metrics

Accuracy

- Defining the rate at which a model has classified the records correctly.

A good model should have a high accuracy score, but having a high accuracy score alone does not guarantee the model is well established. (Necessary, but not sufficient)

True Positive Rate(TPR)

- hows the proportion of positive instances the classifier correctly identified

False Positive Rate(FPR type 1 error rate)

- shows the percentage of negatives the classifier marked as positive

False Negative Rate(FNR type 2 error)

- shows the percent of positives the classifier marked as negatives.

Precision

- the percentage of instances that are actually positive among the marked positives

In General

A well-performed classi□er should have a high TPR (ideally 1) and a low FPR and FNR (ideally 0).

In Practice

...

N-Fold Cross-validation

...