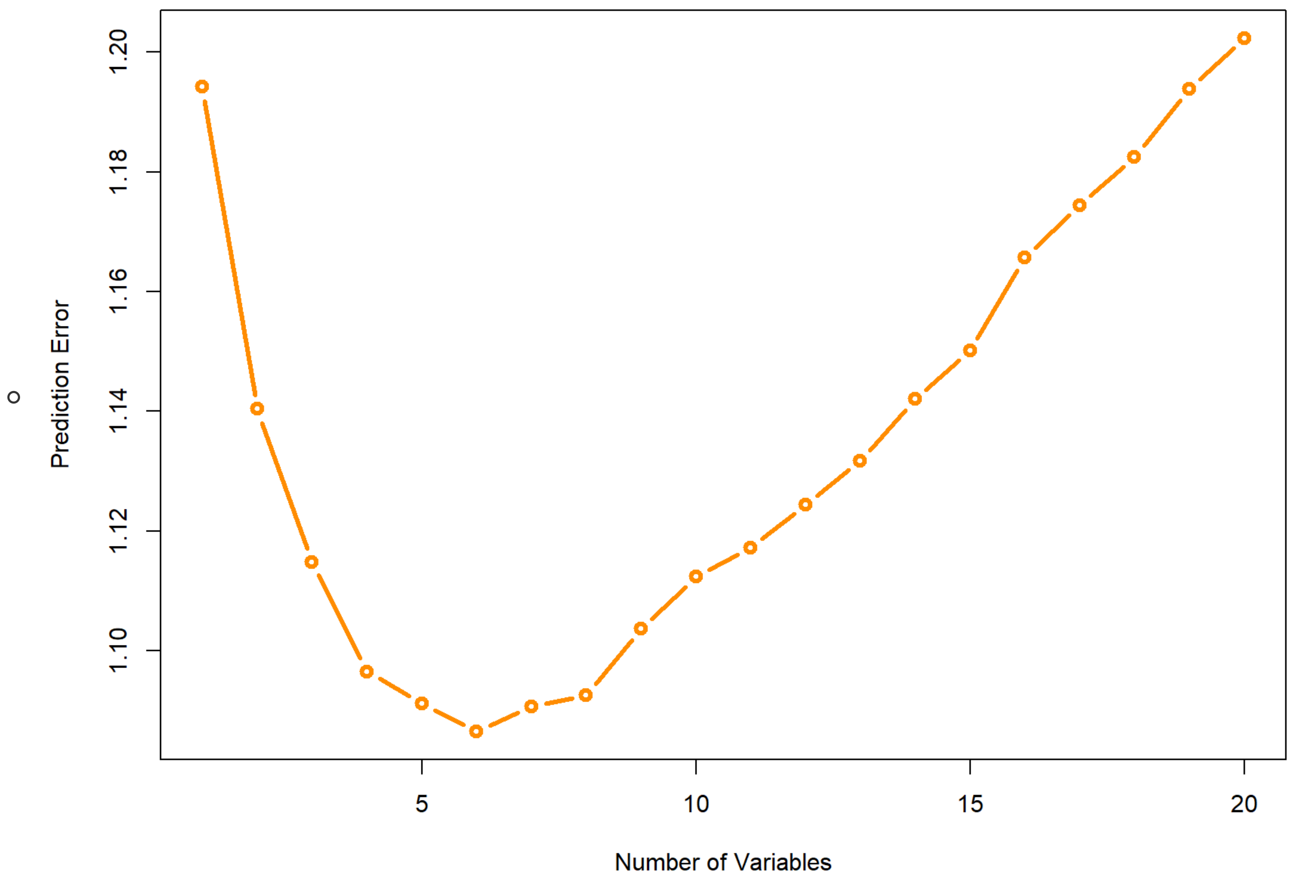
- First order: convex, smooth. 求导令导数=0.
- Second order:

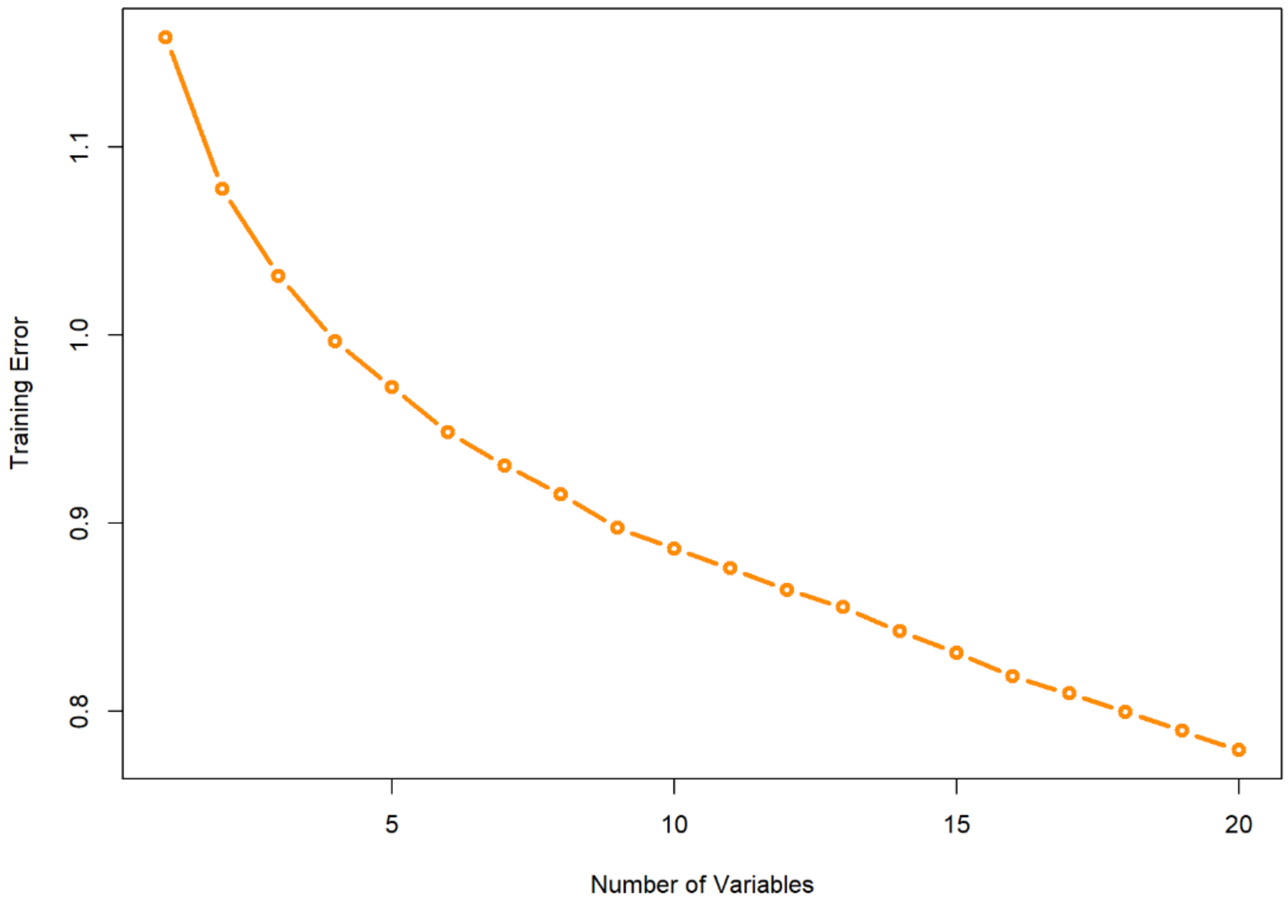
very brief conceptual question on boosting

no question about spline

Week 2

- Model Selection Criteria





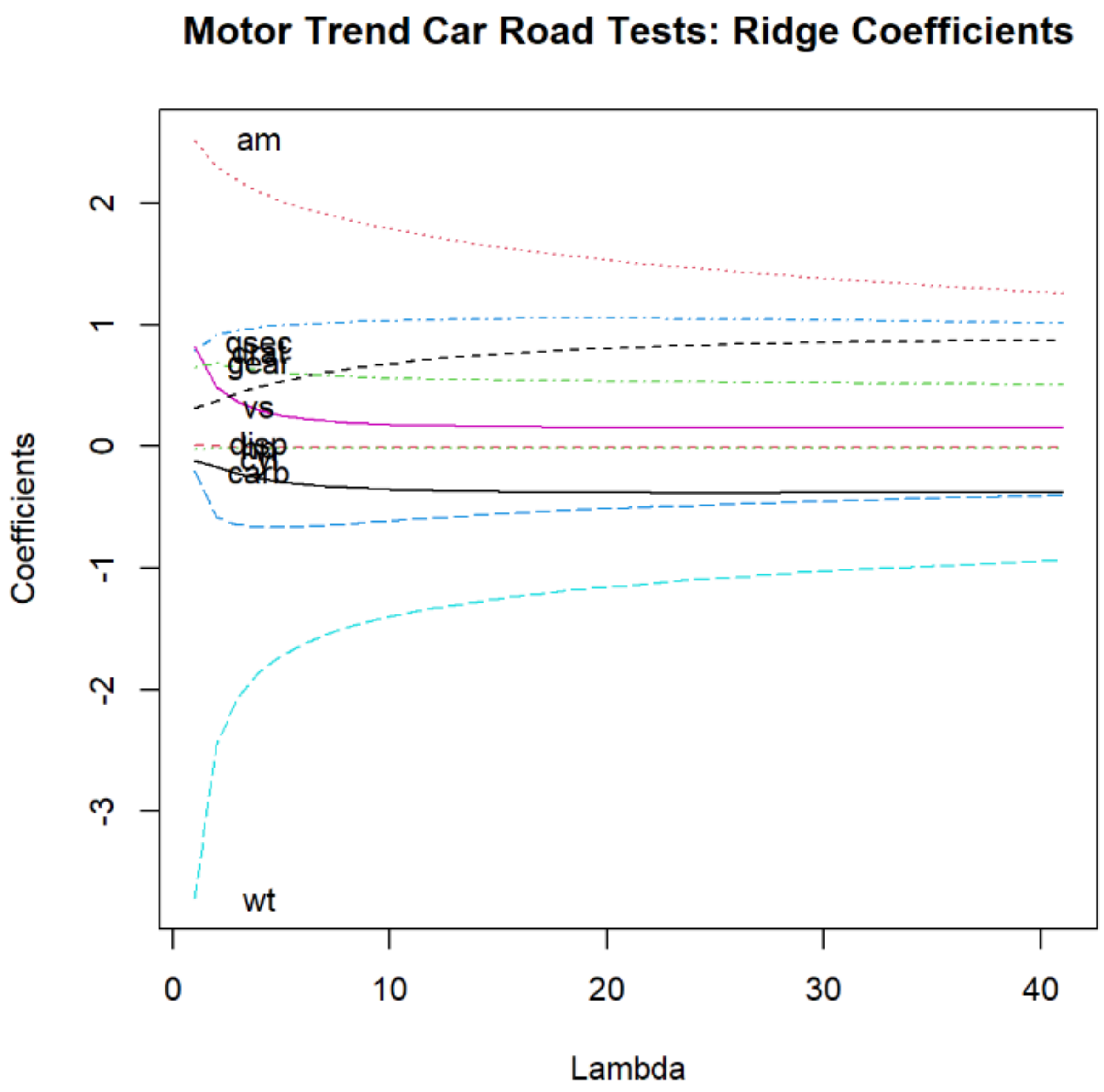
- - Trade-off
    - Training error: 一直减小
    - Prediction error：先减小后增大
  - Marrows' C_p
    - Criterion: minimize RSS + $2p\sigma^2_{full}$
  - AIC and BIC
    - AIC: −2Log-likelihood+2p
    - BIC: −2Log-likelihood+log$(n)p$
  - Best Subset Selection with `leaps`
    - Since penalty is only affect by the number of variables, first choose the best model witht the smallest RSS for each model size. Compare by attaching the penalty terms of their correspoonding sizes.
    - calculate teh best mdoel of each mdoel size.
  - Step-wise regression using `step()`

- Numerical Optimization: Basic Concepts

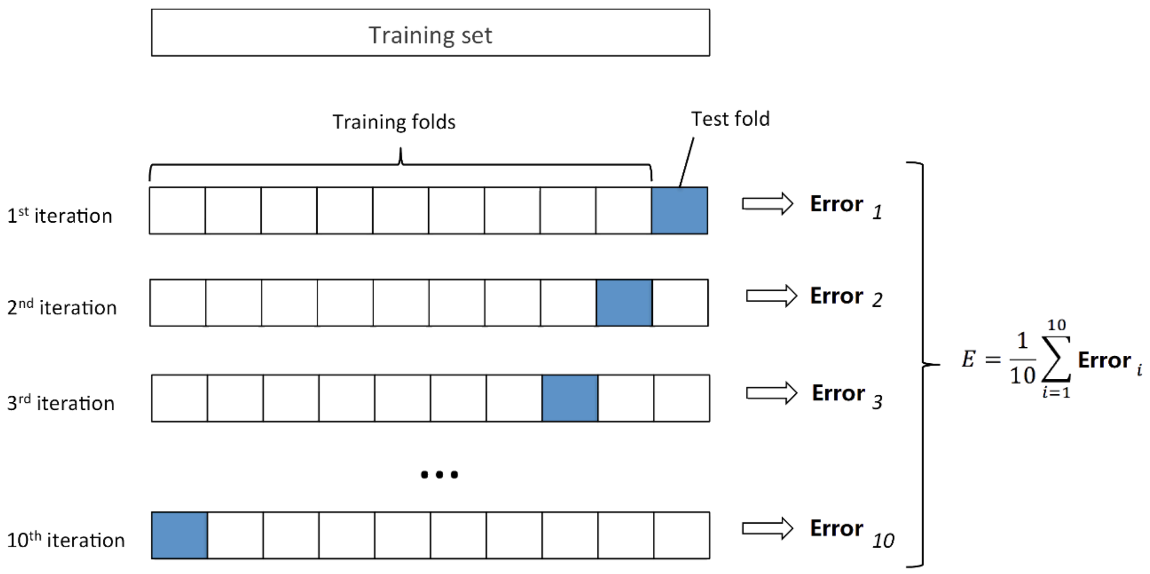
    - `optim()` function
    - first order. Gradient descent. Convex function 上凸函数，有最低点
    - second property. Hessian matrix >= 0
    - Newton's method

Week 3 Ridge

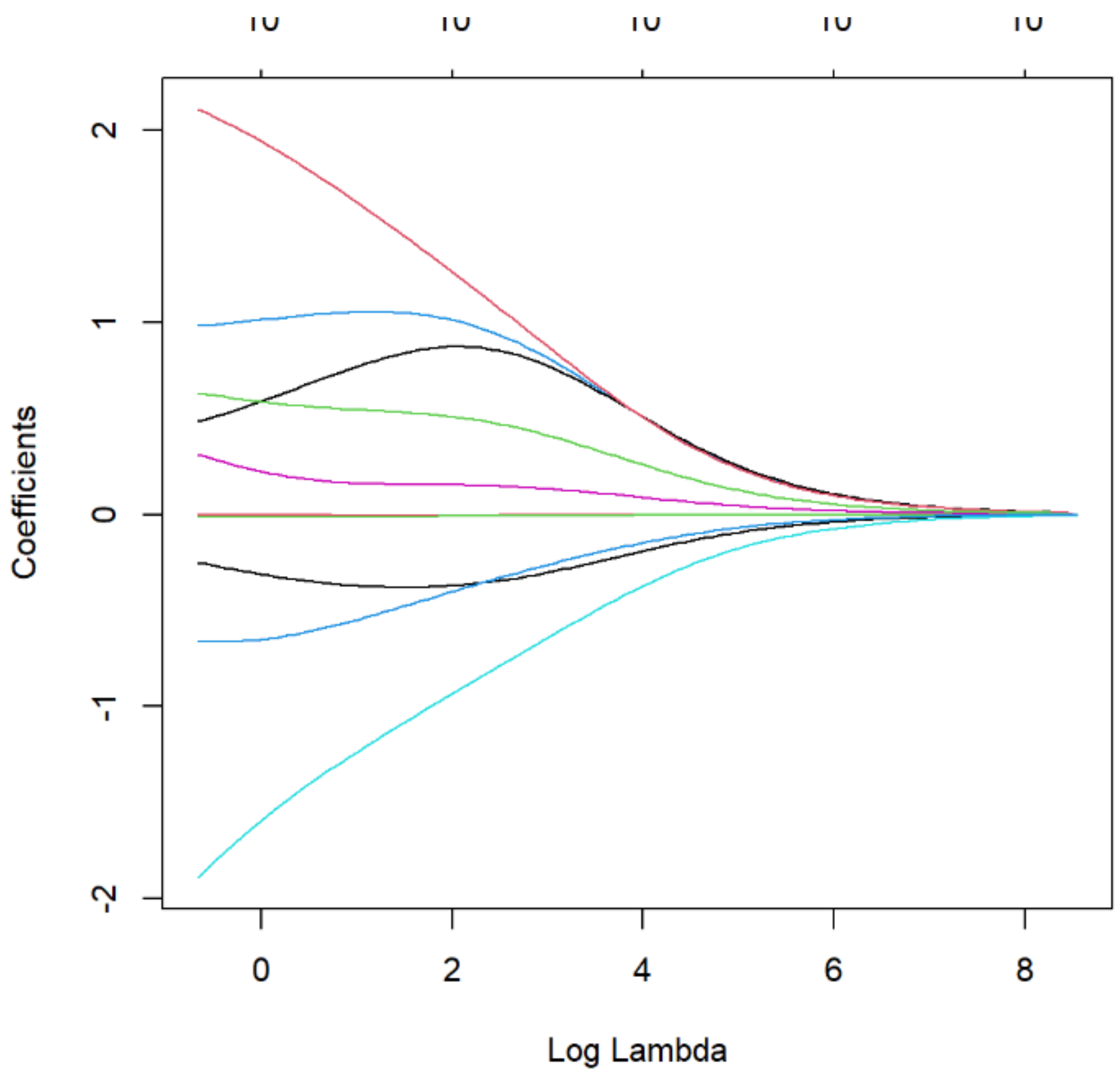
- Motivation
    - deal with highly correlated variables
    - Solution is unstable since the optimizer could land anywhere
- Ridge penalty--reduced variance
    - Ridge $l2$ penalty
    - more stable
        - objective function is more convex and less affected by the random smaples
        - Var of the estimator is smaller because the eigenvalues of X^TX + nλI are large.
- A biased estimator--Bias caused by the Ridge penalty
    - OLS estimator--unbiased
    - Compare β^{ridge} and β^{OLS}, OLS--unbiased, ridge--biased, but the varition for β^{ridge} is much less than the one for β^{OLS}
- Bias-variance Trade-off
    - Penalization leads to a biased estimator(since OLS estimator is unbiased)
        - As λ→0 the ridge solution is eventually the same as OLS
        - As λ→∞, β^{ridge}->0
    - As λ↓λ↓ decrease, bias ↓↓ decrease and variance ↑↑ increases
    - As λ↑λ↑ increases, bias ↑↑ increases and variance ↓↓ decrease
    - Overall effect : $Bias^2$ + Variance + irreducible error
- Using `lm.ridge()`
    -



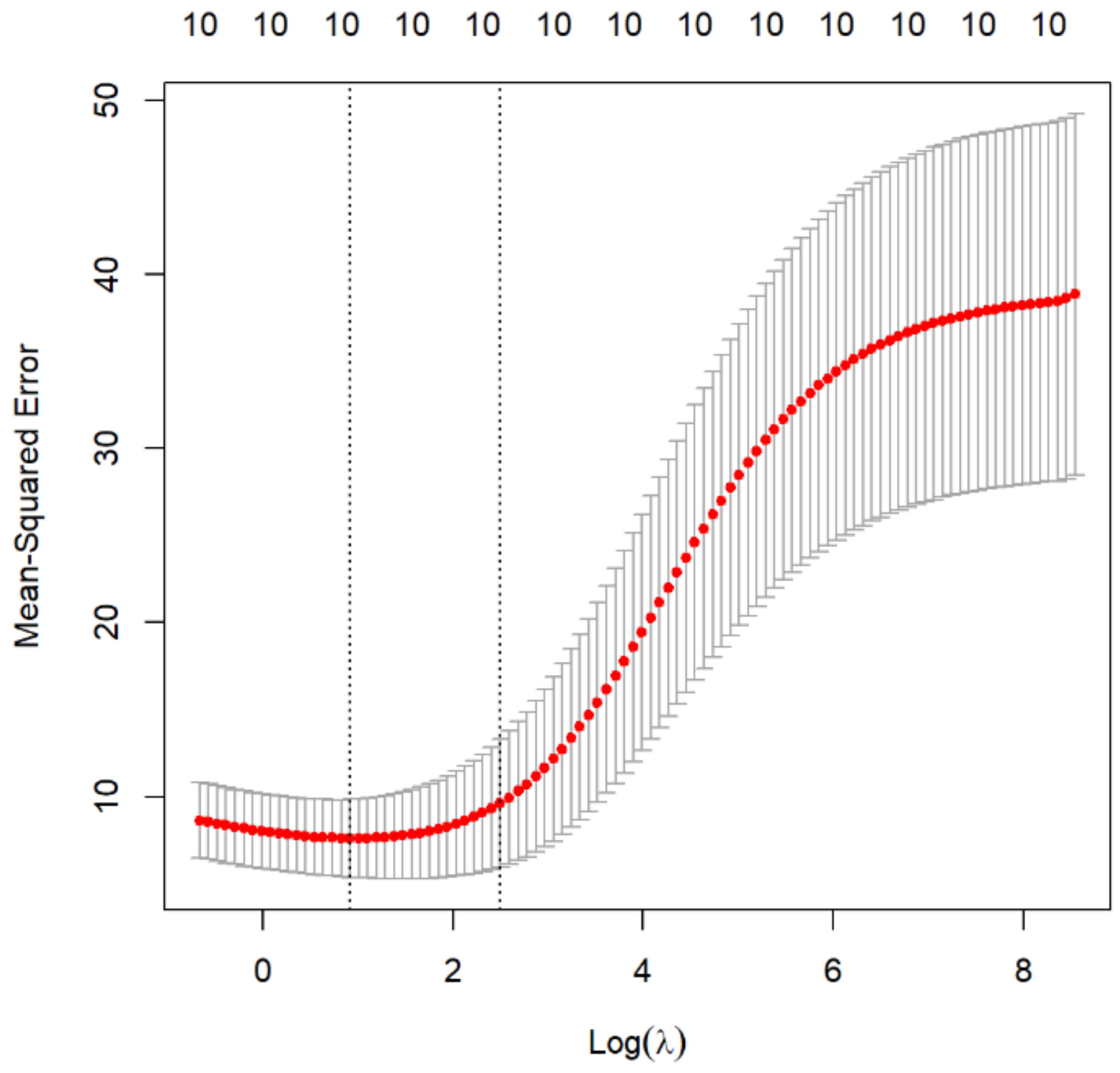
- Select best λ(many methods, talk k-fold CV here)
  - K-fold CV
    - Randomly split the data into KK equal portions
    - For each kk in $1,...,K$1,...,K: use the $k$kth portion as the testing data and the rest as training data, obtain the testing error
    - Average all K testing errors
    -



- `glmnet` package
  - implement k-fold CV
  - alpha = 0
  -



- s = "lambda.min"`s = "lambda.min"`, and sometimes practitioners use `s = "lambda.1se"` to select a slightly heavier penalized version based on the variations observed from different folds.
-



- Scaling Issues
  - Both the `lm.ridge()` and `cv.glmnet()` functions will produce a ridge regression solution different from our own naive code. This is because they will internally scale all covariates to standard deviation 1 before using the ridge regression formula. The main reason of this is that the bias term will be affected by the scale of the parameter, causing variables with larger variation to be affected less. However, this scale can be arbitrary, and such side effects should be removed. Hence both methods will perform the standardization first.

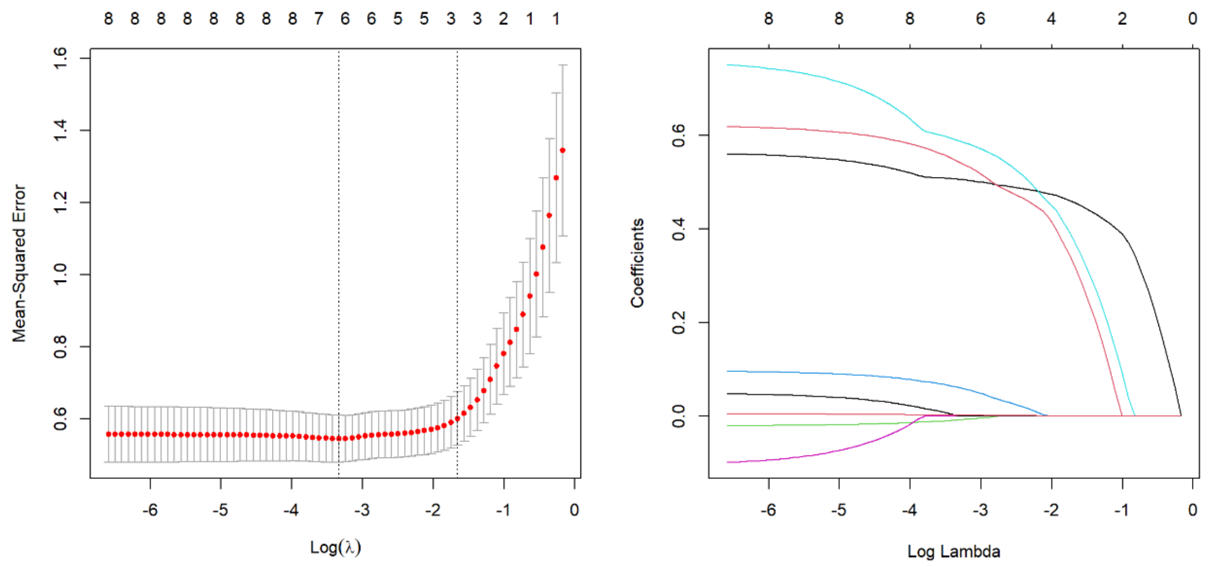
Week 4

- Lasso
  - add $l1$ penalty on the fitted parameters
    -

$$\widehat{\boldsymbol{\beta}}^{\text{lasso}} = \arg\min_{\boldsymbol{\beta}} \|\mathbf{y} - \mathbf{X}\boldsymbol{\beta}\|_2^2 + n\lambda\|\boldsymbol{\beta}\|_1$$

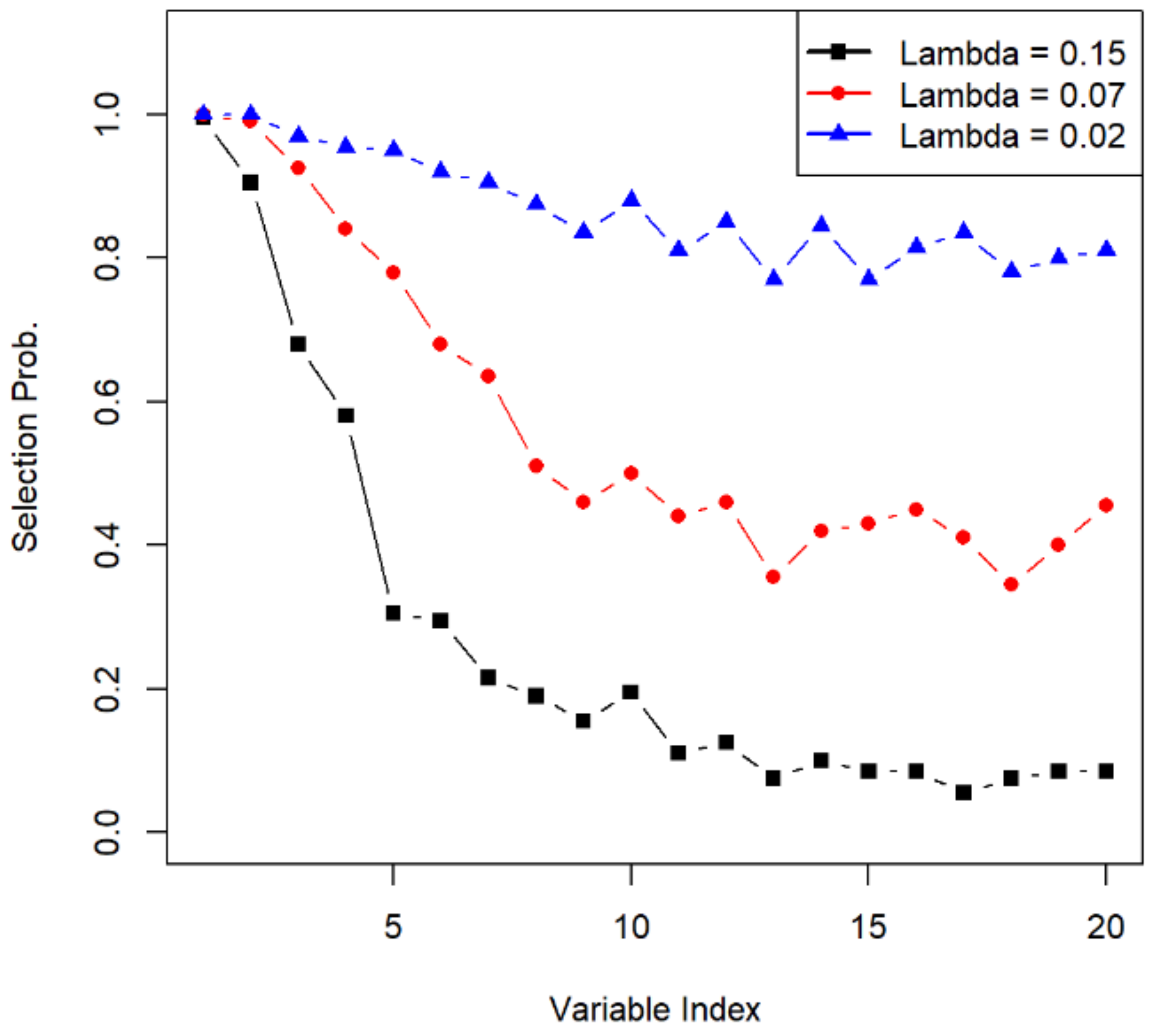
$$= \arg\min_{\boldsymbol{\beta}} \frac{1}{n}\sum_{i=1}^{n}(y_i - x_i^{\mathrm{T}}\boldsymbol{\beta})^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

- Advantage: small β_j can be shrunk to zero("varaible selection property")
    - sparse model, prevent over-fitting
    - improve the interpretability when dataset is large

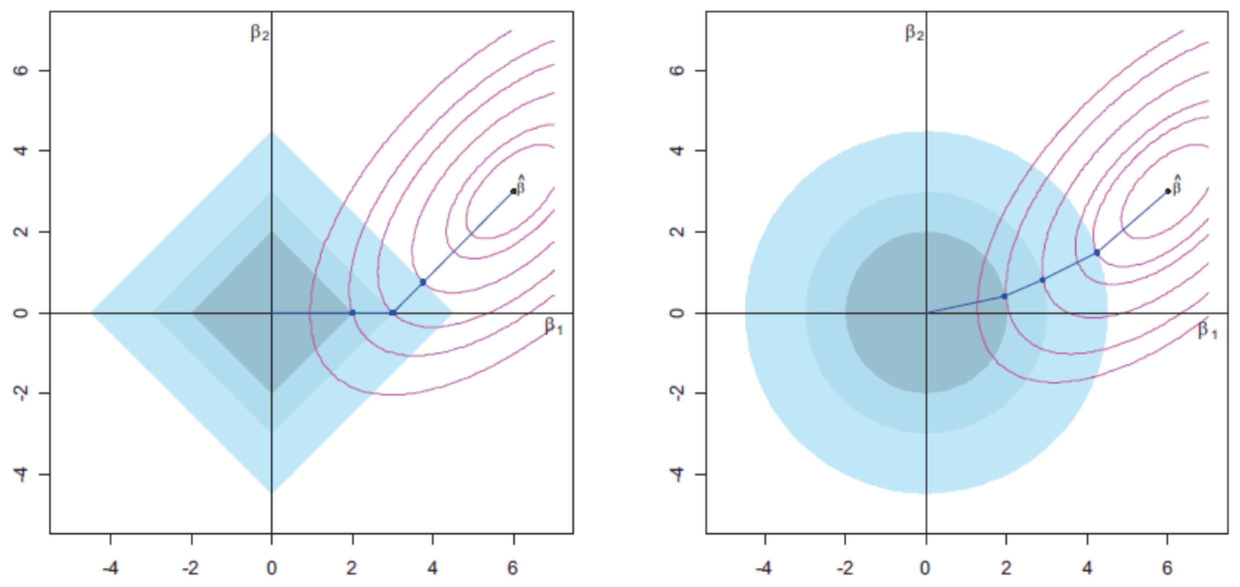
- Using `glmnet` package
    - alpha = 1
    - Two vertical lines, represents `lambda.min` and `lambda.1se` respectively.
        - `lambda.min`: minimize the CV error, contains more nonzero parameters, since a larger penalty $lambda$ will force more parameters to be zero, so the model is more sparse
        - `lambda.1se`: give larger penalty value with more shrinkage
    - 

    - in the right graph, when λ becomes larger, most parramters estimates stays at 0, model becomes sparse

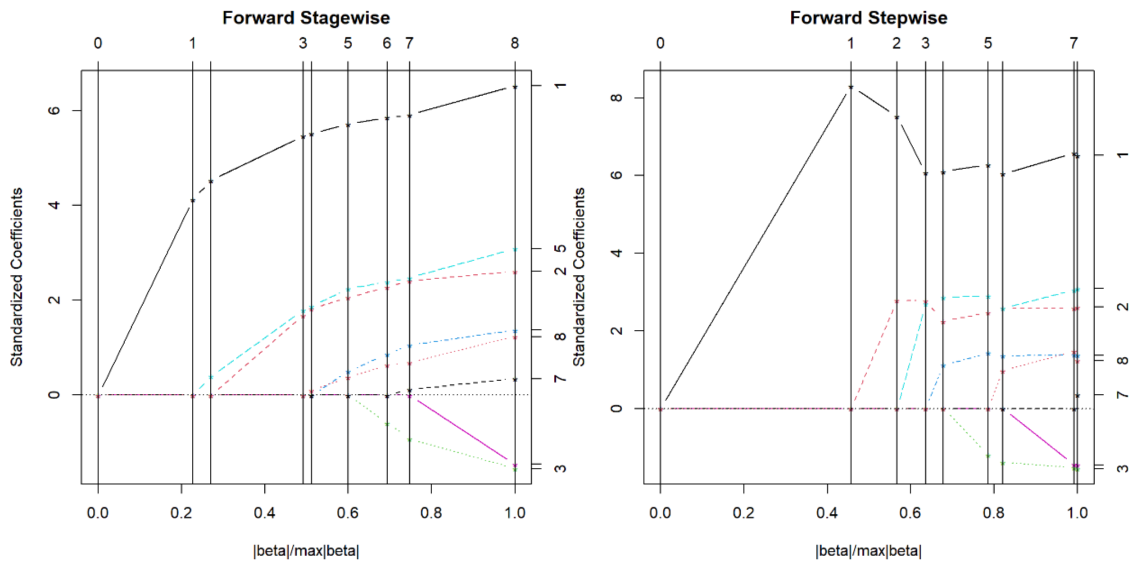
- One-variable Lasso and shrinkage
    - ...
    -



- Constraind Optimization View
    - Once the contained area is sufficiently small, some ββ parameter will be shrunk to exactly zero
    - since the constrained areas is a circle, it will never for the estimated parameters to be zero.
    -



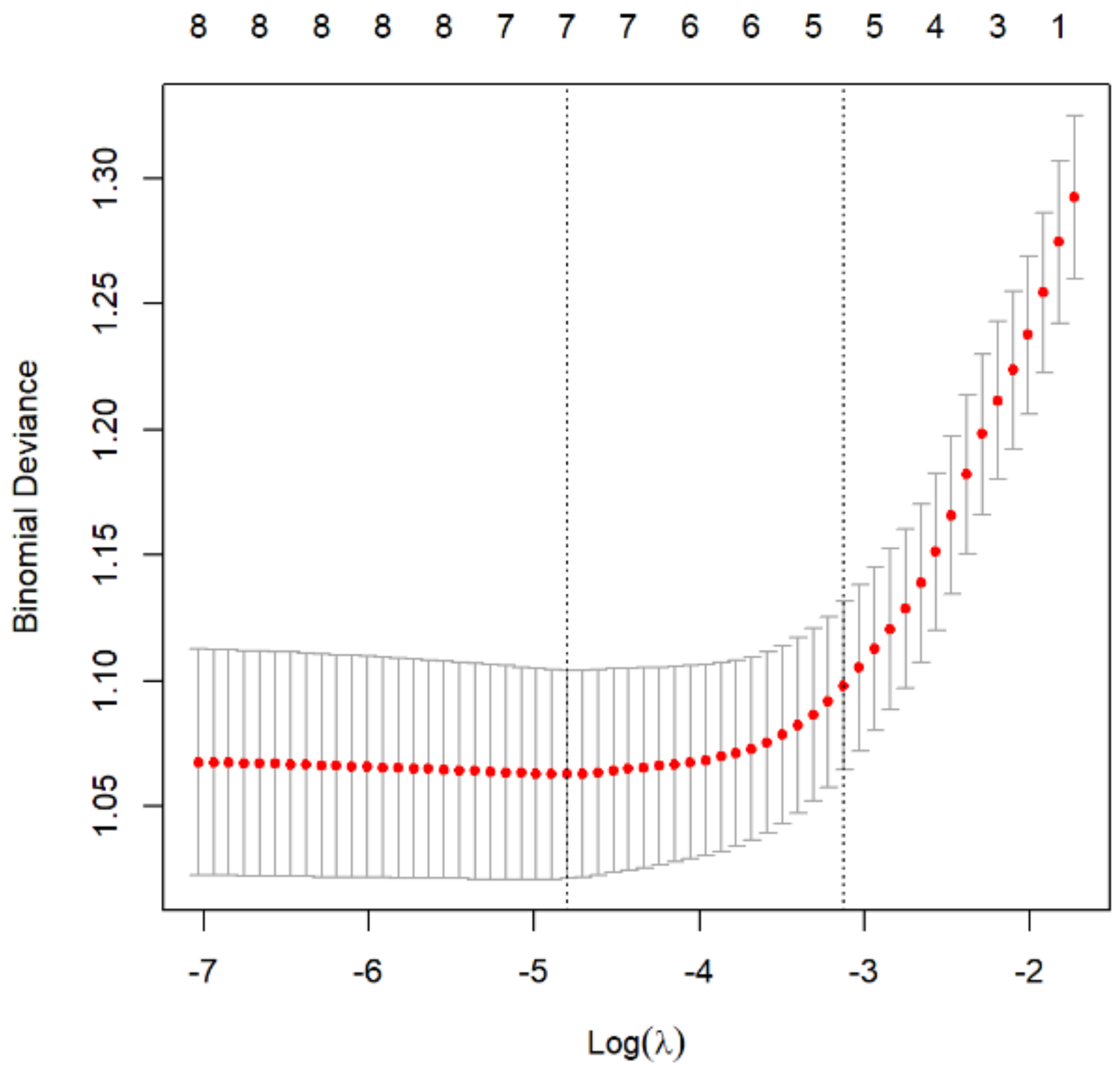
- Bias-variance trade-off
  - As $\lambda\downarrow$decrease, bias $\downarrow$ decrease and variance $\uparrow$ increases
  - As $\lambda\uparrow$ increases, bias $\uparrow$ increases and variance $\downarrow$ decrease
- Forward-stagewise Regression
  - Select tuning parameter λ, select on a sequence of λ values. Correspondinig β parameter estimates are called **solution path**. For Lasso, the solution path has an interpretation as the **forward-stagewise** regression.
    - RSS reduce the most
    - Forward Stagewise vs. Forward Stepwise
    -



    - different becasue the parameters are presented in the scale as if their standard deviations are all 1
- Elastic-Net
  - motivation: lasso suffer when two variables are strongly correlated. (Lasso will only select one

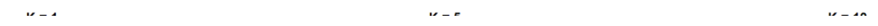
        - motivation: lasso suffer when two variables are strongly correlated. (Lasso will only select one out of the two)
        - **Elastic-Net** is used when the data contains many correlated variables and we want to select them together if they are important for prediction.

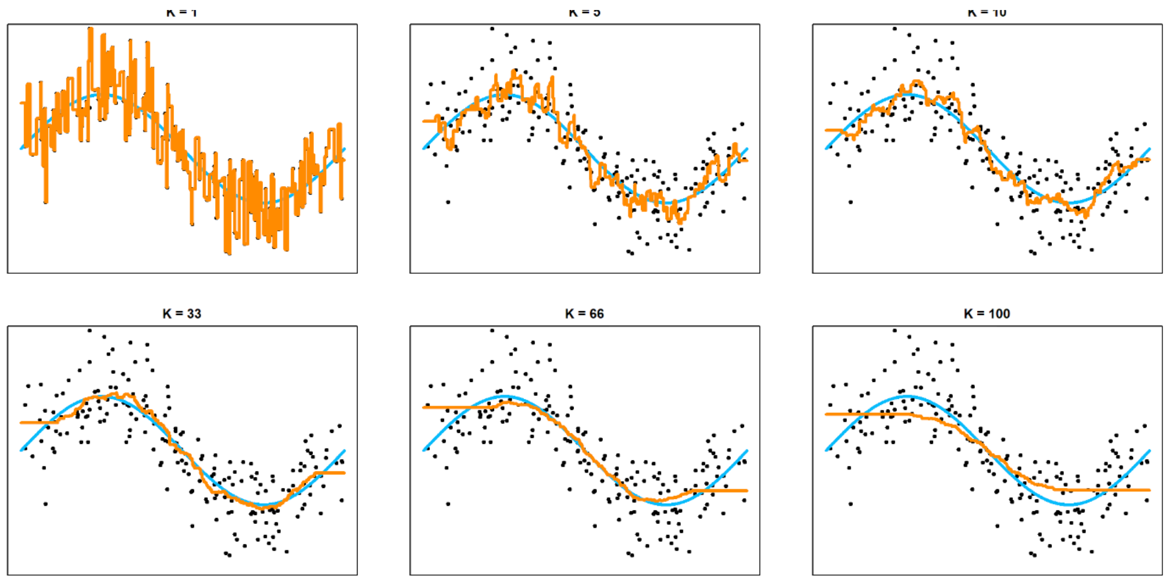
- Logistic Regression
    - binary outcome, Y ~ 0 or 1
    - Example of **Generalized Linear Models** (when X xchanges, probability always be bounded by 0 and 1)
    - Relationship represented using a **logit link** function
    - $$\eta(a) = \frac{\exp(a)}{1 + \exp(a)}$$
    - **log odds** = $x^T \beta$
    - larger than 0.5--1, smaller than 0.5 -- 0
    - Interpretation of parameters
        - log(odds ratio) = β_1, the parameter β of a varaible in a logistic regression represents the **log of odds ratio** associated with one-unit increase of this variable.
    - Solving
        - Gradient/coordinate descent are both fine
        - can use Newton's method to update
        - Key: derive the gradient and Hessian functions
    - Penalized Logistic Regression
        - similiar to linear regression, apply penalties to:
            - address collinearity problems
            - select variables in high-dimentional setting
            - can add lasso/Ridge penalty

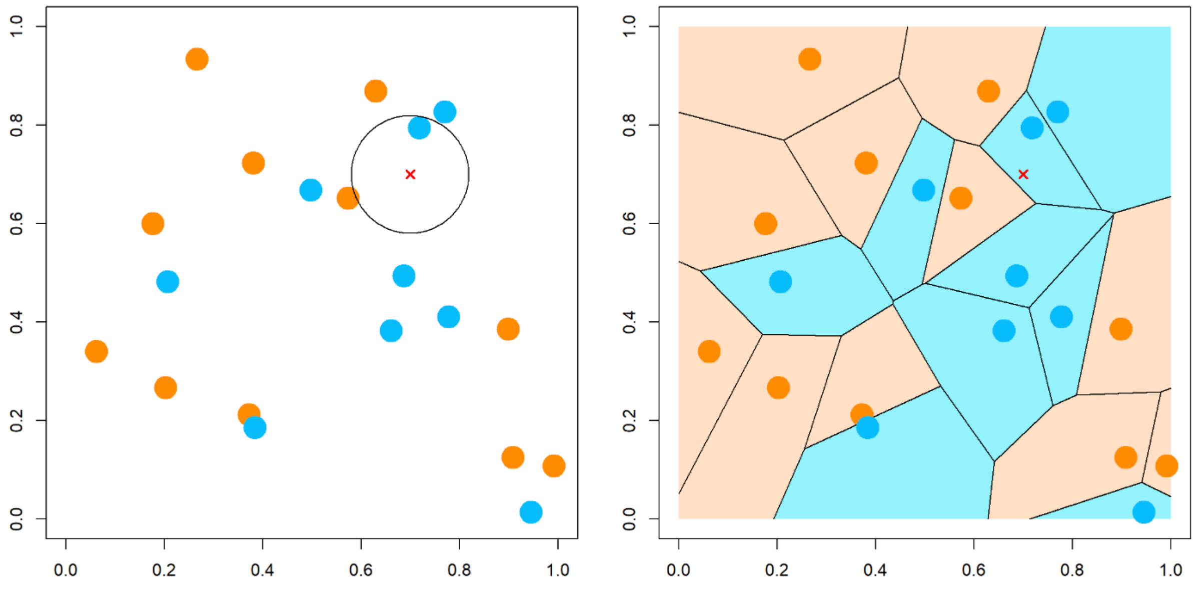


-
    - select best λ(Lambda.min)

Week 5

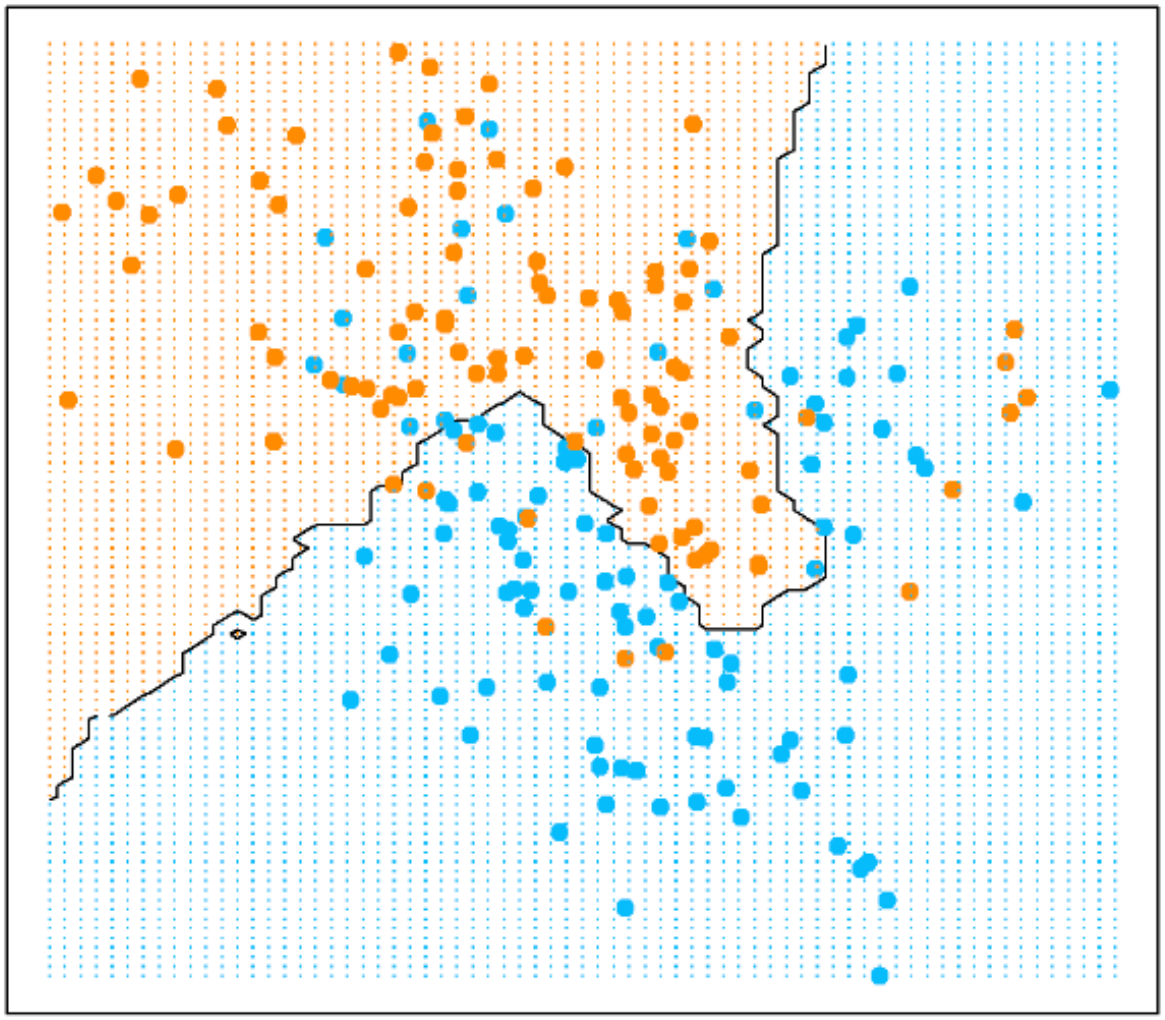
- KNN
  - nonparametric method(direct estimate $f(x_0)$, not have parameters like $beta$, use local averaging techinique), for both regression and classification problems.
  - KNN for regression
    - Tuning k
      - 



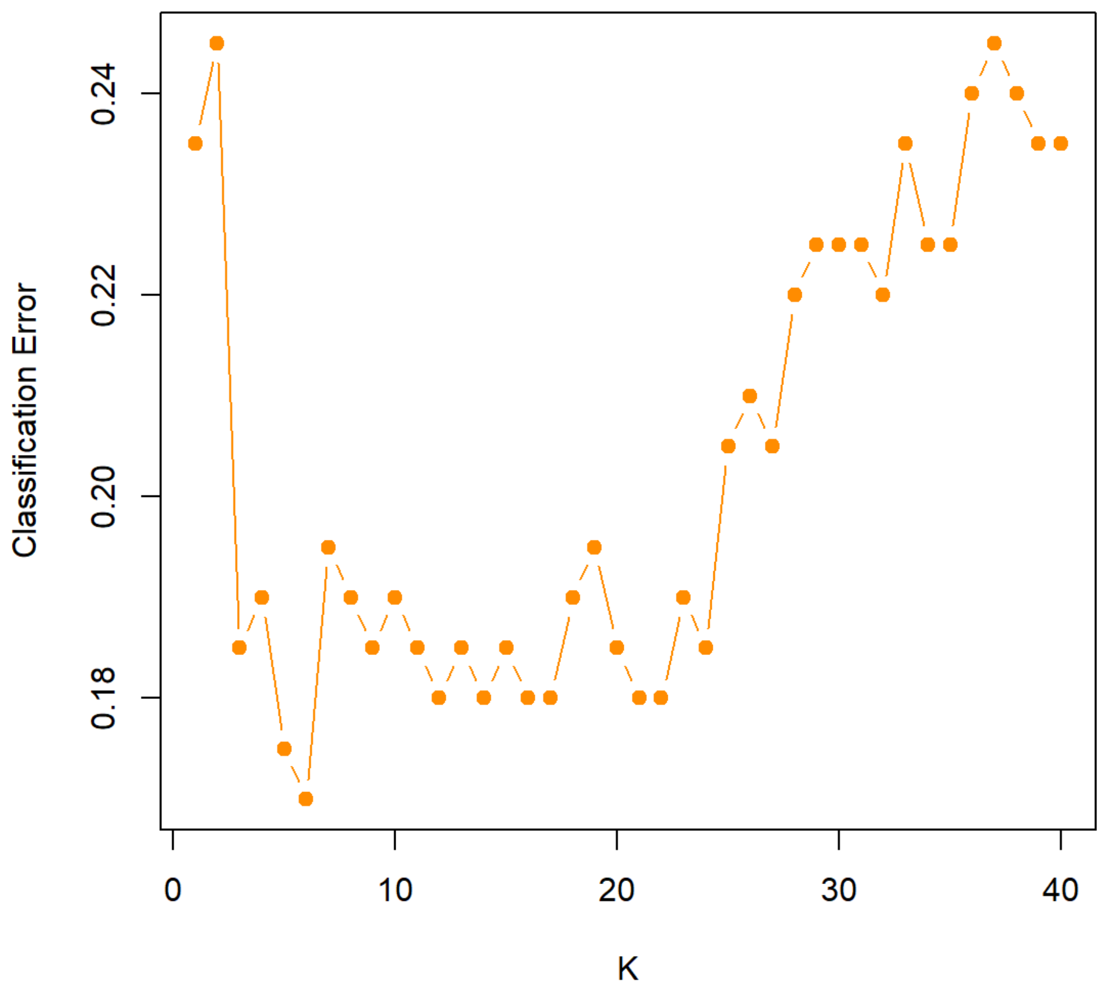
    - k increases, smaller var, larger bias
  - Bias-variance tradeoff
    - prediction error = irreducible error + $Bias^2$ + Variance
    - As k increases, bias increases and variance decreases
    - As k decreases, bias decreases and variance increases
- KNN for classification
  - different from regression when finding neighbors: majority voting instead of averaging.
  - 

  -

15-Nearest Neighbor



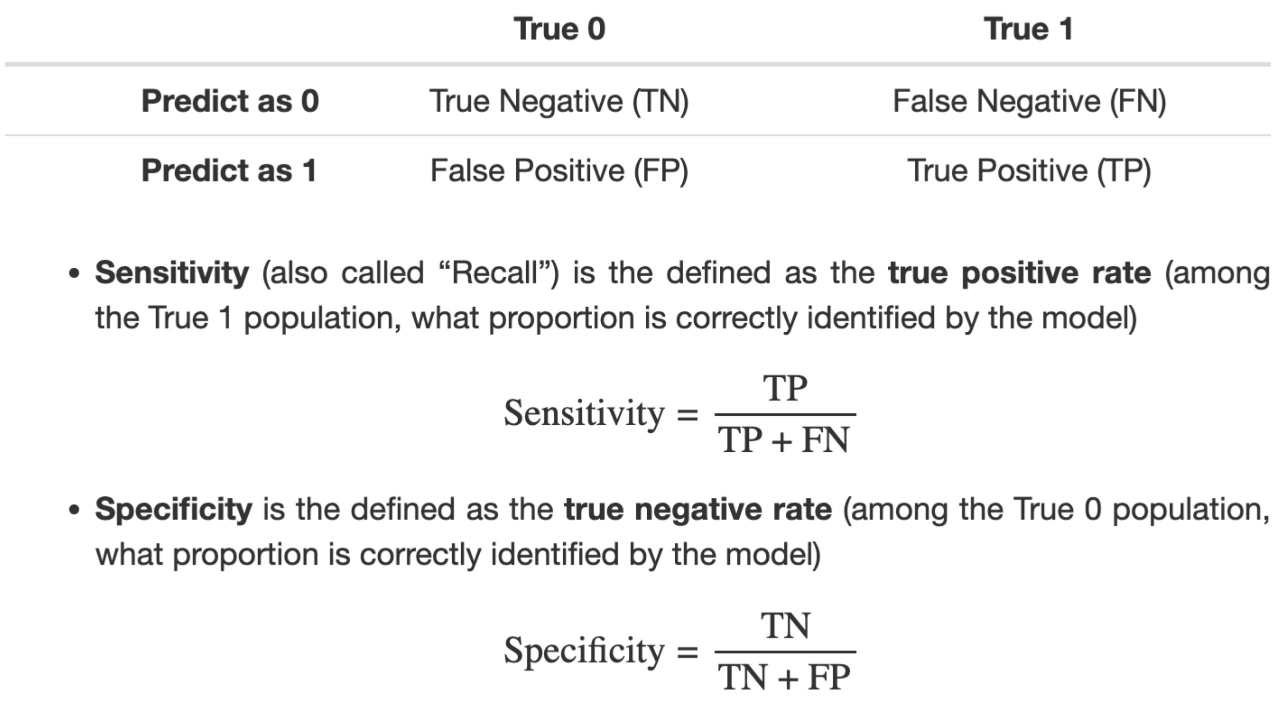
- Tuning with `caaret` Package
    -



    - best k is 6
    - Clear "U" shape, shows tradeoff
  - Computational Issues
    - high computational cost and not memory friendly for sample sample size
    - to find the nearest neighbors, we need to calculate and compare the distances to each of the data point
    - need to store the entire training dataset for future prediction(only need to store $beta$ for linear model)

- Curse of Dimensionality
  - Motivation: high dim data, fix sample size n, p increases, data becomes very sparse.. KNN doesn't perform well on high-dim dataset. (reason: for given target point, no enough training data lies close enough.)
  - why does our model perform well in the handwritten digit example? There is possibly (approximately)

a lower dimensional representation of the data so that when you evaluate the distance on the high-dimensional space.

- Classification Model Evaluation
    - Previously, using classificationerror
        - fine for most application
        - problem with unbalanced dataset(for logitic regression)
    - we need
        - use difference cutoff values
        - new measurements other than prediction error
    - 

	True 0	True 1
Predict as 0	True Negative (TN)	False Negative (FN)
Predict as 1	False Positive (FP)	True Positive (TP)

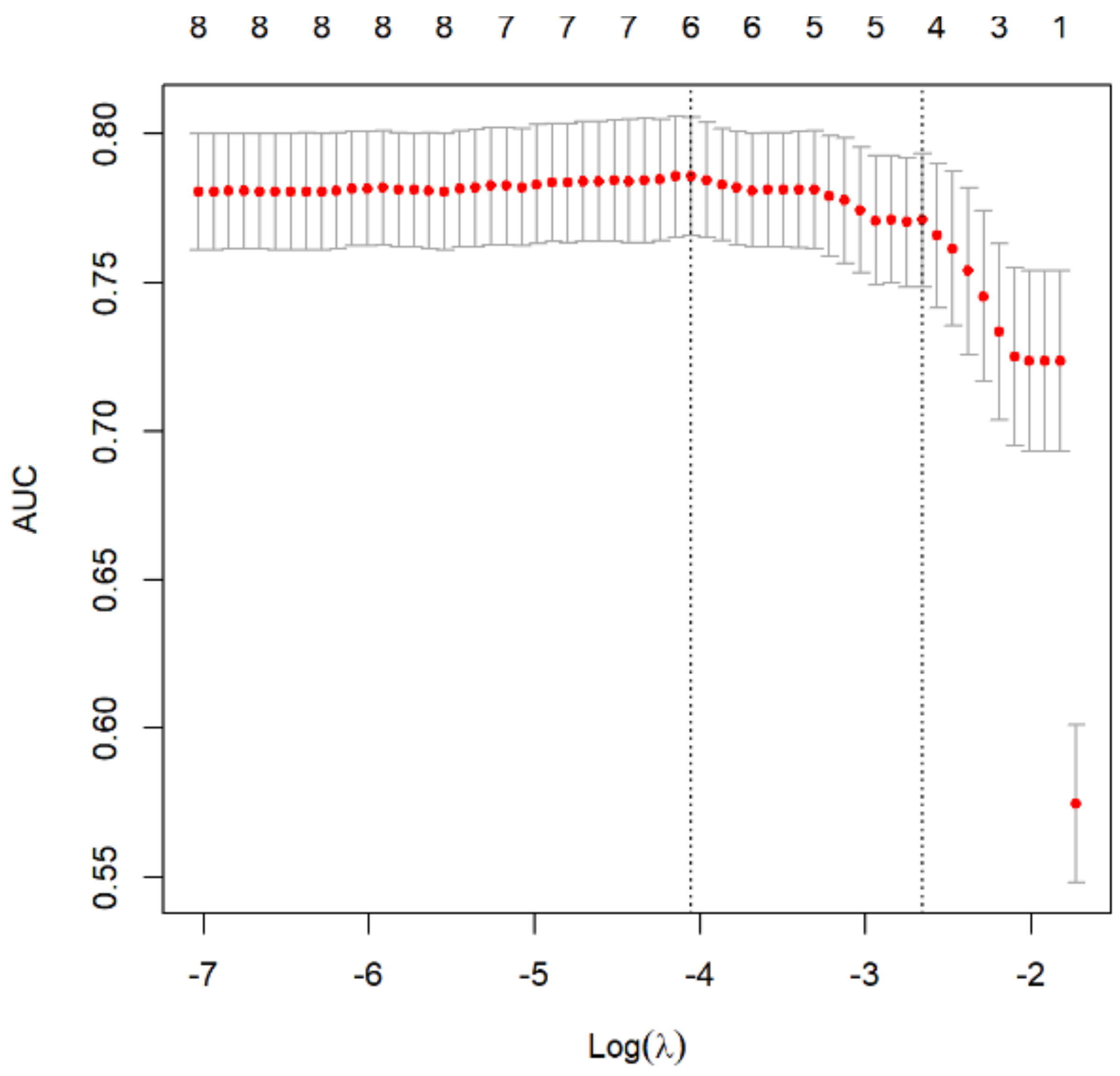
- **Sensitivity** (also called "Recall") is the defined as the **true positive rate** (among the True 1 population, what proportion is correctly identified by the model)

$$\text{Sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **Specificity** is the defined as the **true negative rate** (among the True 0 population, what proportion is correctly identified by the model)

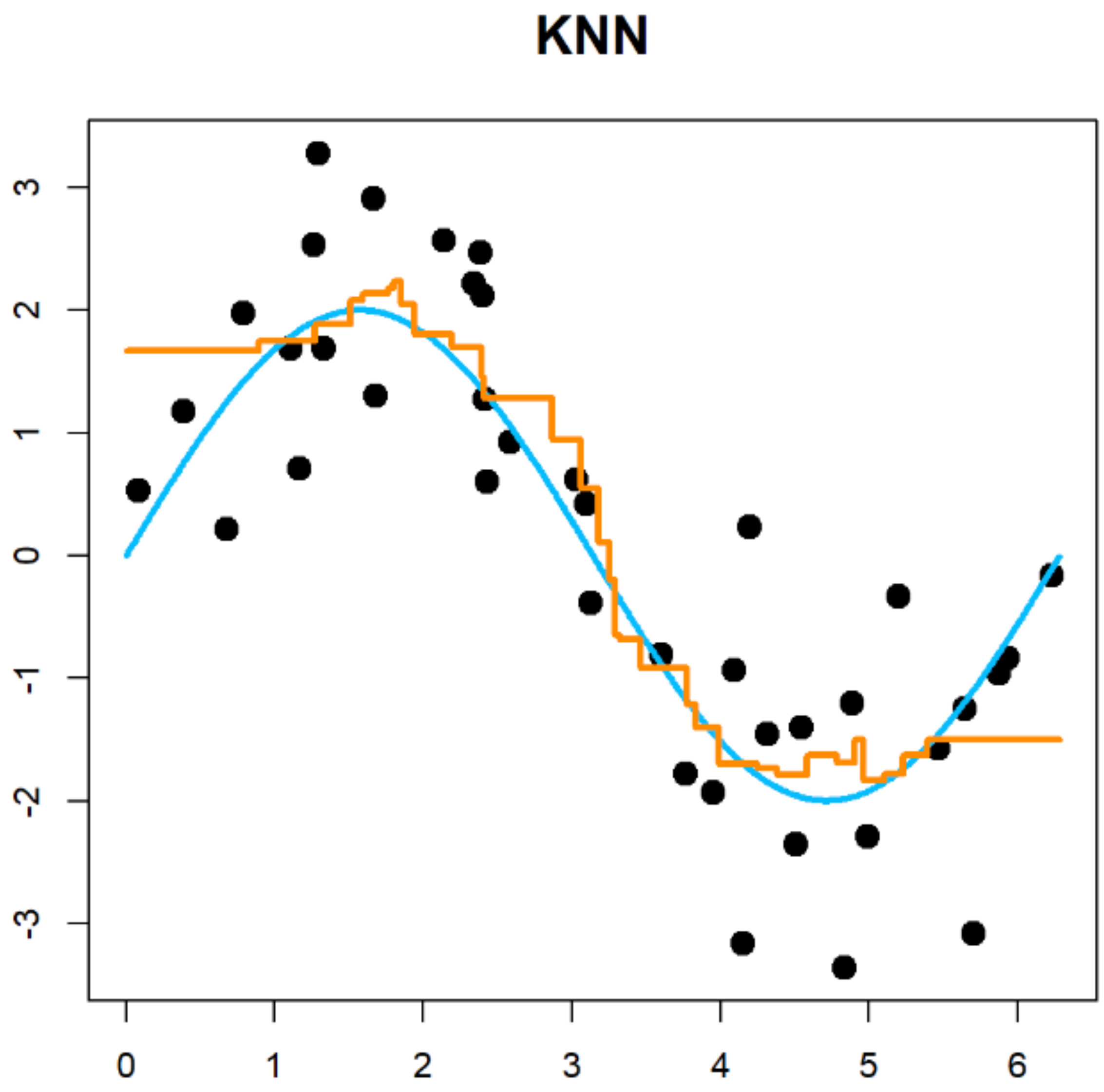
$$\text{Specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

    - ROC curve(`1 - specificity` (false positive rate) versus the `sensitivity` (true positive rate).)
    - AUC(area under the ROC curve).
        - CV using AUC: larger AUC is better. Pick $/lambda$ gives the largerest AUC, still called `lambda min`.
        -

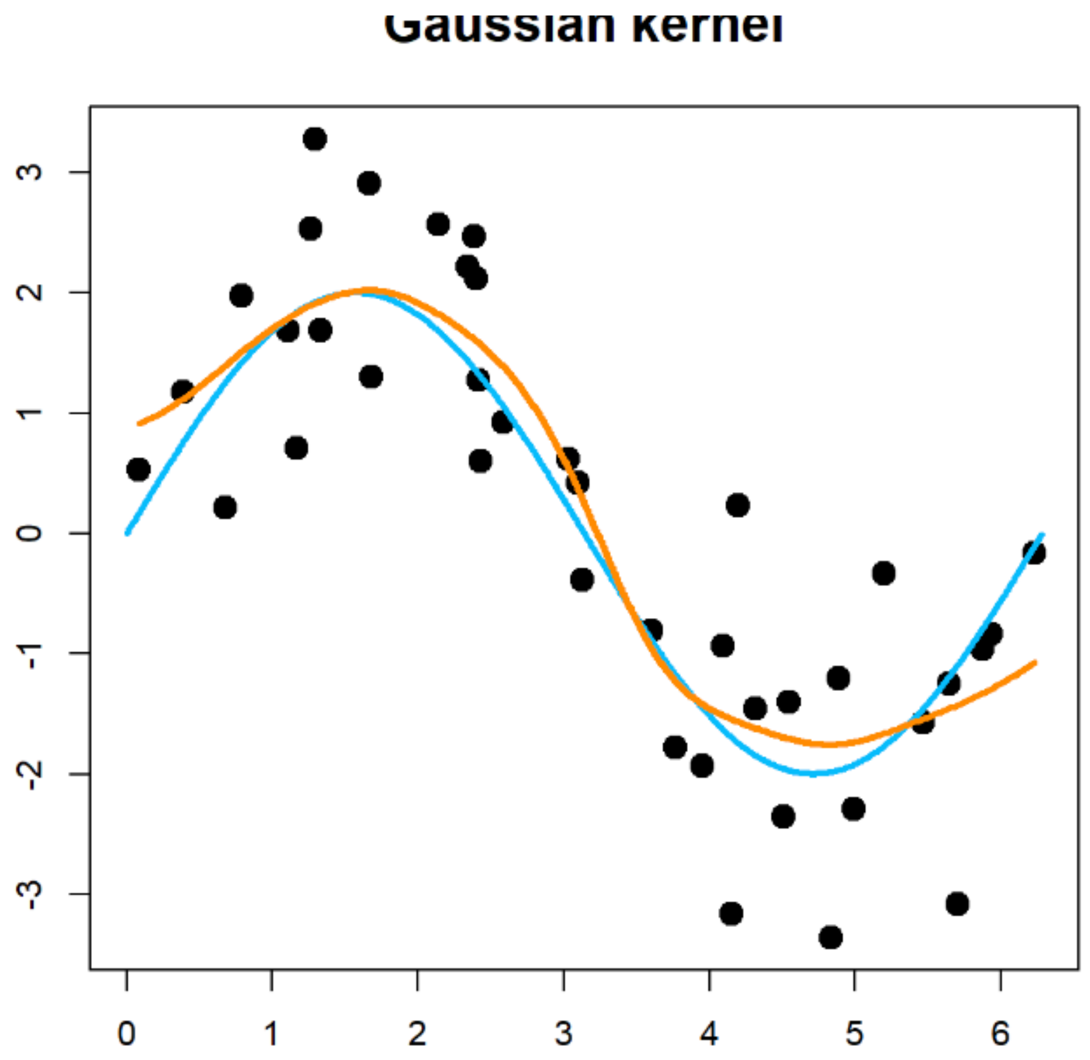


Week 6

- Kernel Smoothing
  - Nadaraya-Watson kernel regression: a new local smoothing estimator, tuning parameter h(called the `bancwidth`), kernel weight K
  - $$\widehat{f}(x) = \frac{\sum_i K_h(x, x_i) y_i}{\sum_i K_h(x, x_i)}$$
  - Comparation
    - KNN: jumps

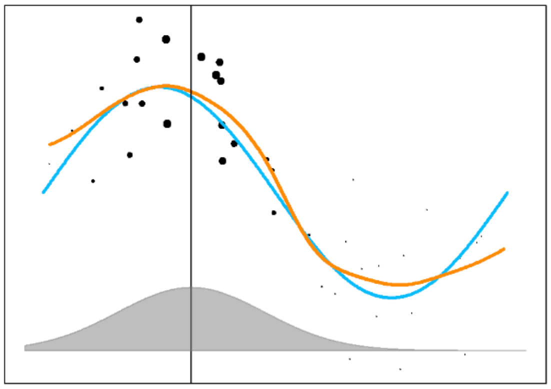
-
  - 

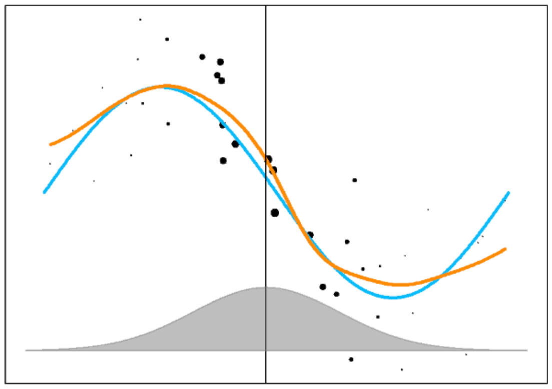
- Kernel regression: smooth(even when increase k)
  -



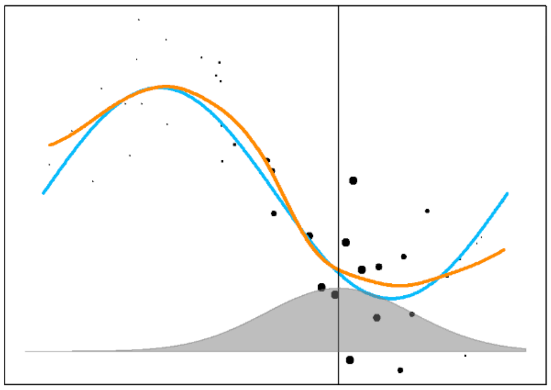
- The local average
  - Training data x_i that are closer to x receives higher weights $K_h(x, x_i)$, hence their $y_i$yi values are more influential in terms of estimating $f(x)$.
  -

Kernel average at x = 2 **Kernel average at x = 3**

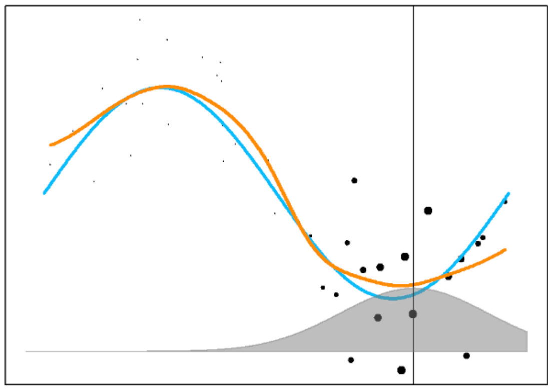




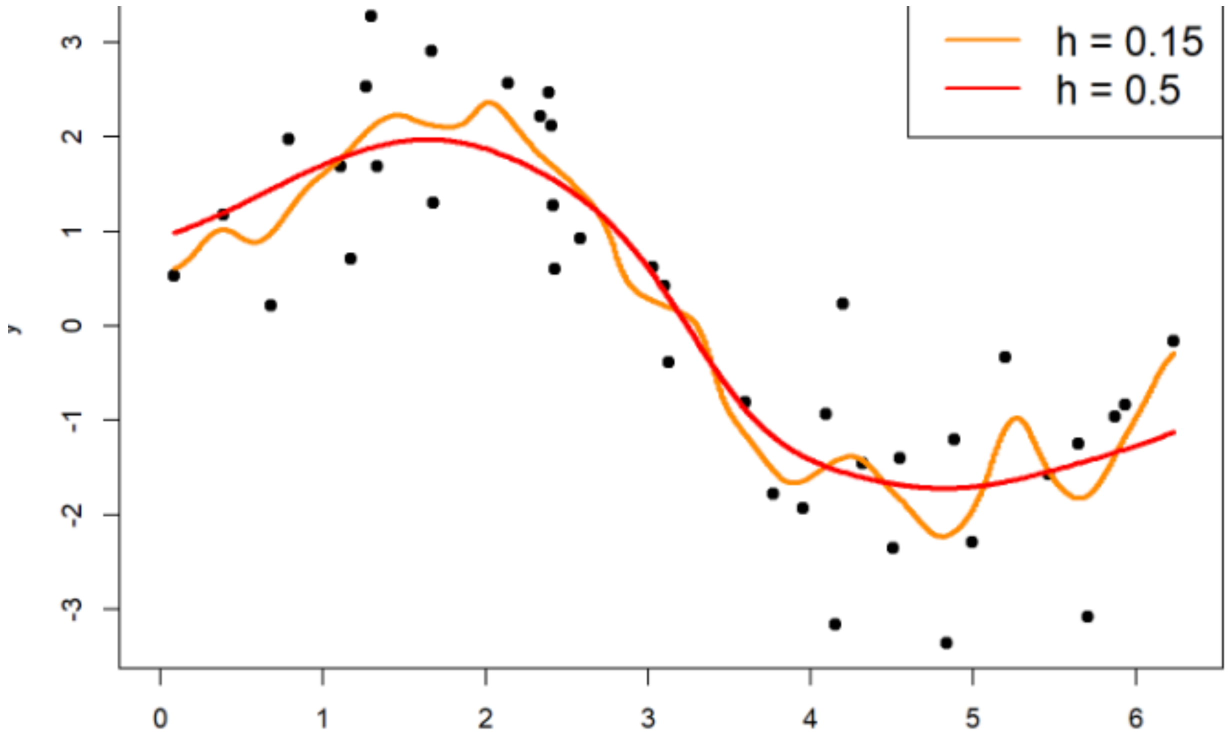
Kernel average at x = 4



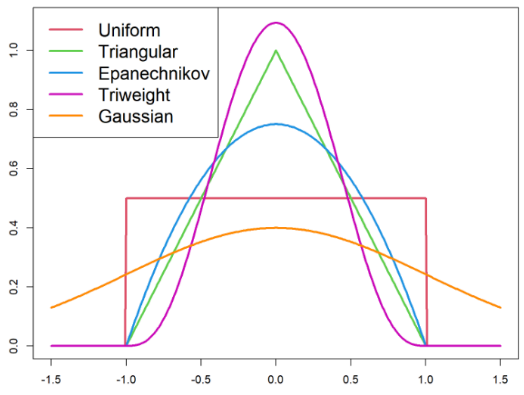
Kernel average at x = 5



- Trade-off
  - tuning parameter: bandwidth h
  - similiar to KNN: larger h, smaller var, larger bias(using larger neighborhood)
  -



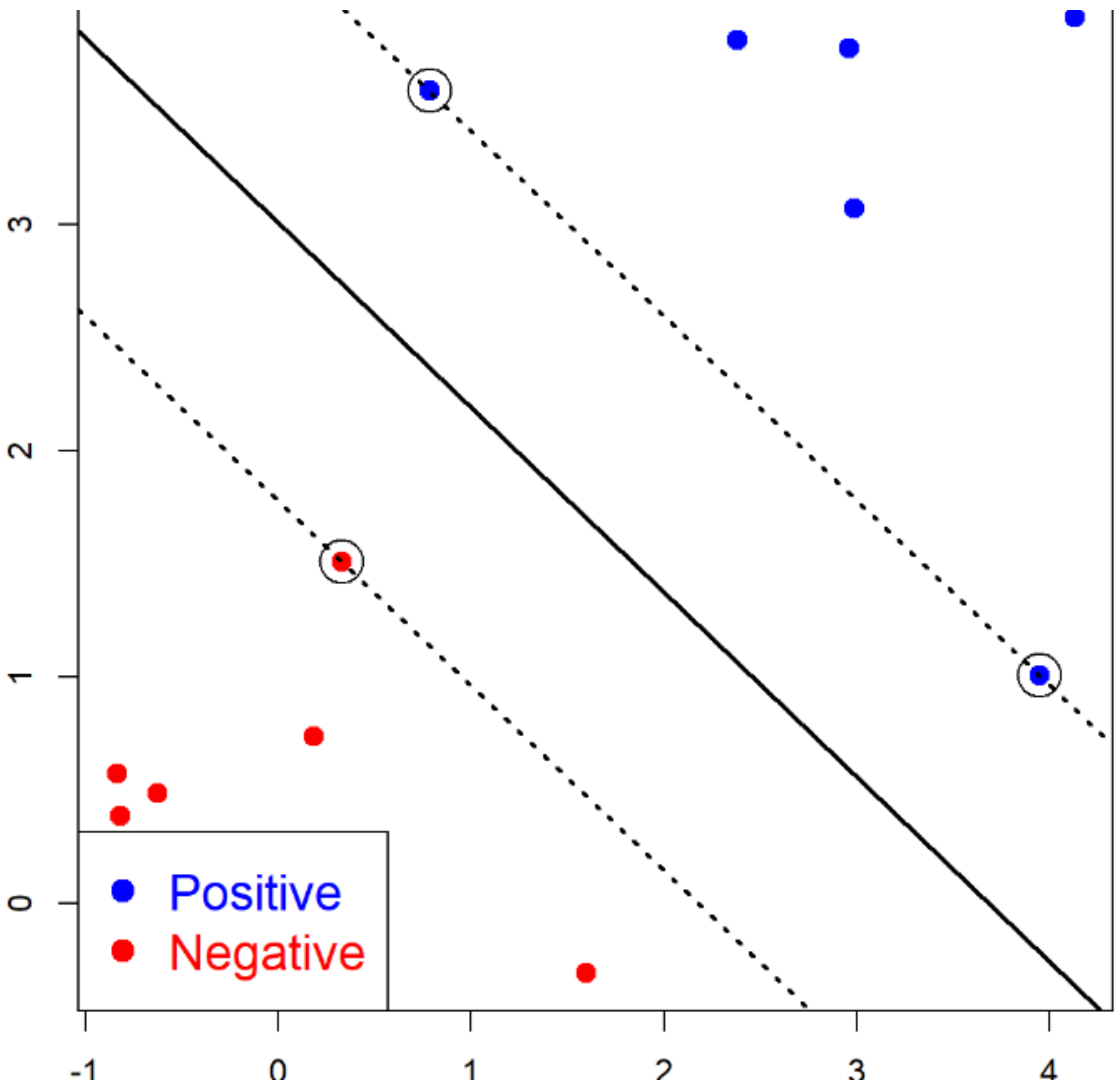
- Kernel Functions
  - Other kernel functions can also be used. The most efficient kernel is the Epanechnikov kernel. Different kernel functions can be visualized in the following. Most kernels are bounded within $[-h/2, h/2]$, except the Gaussian kernel.



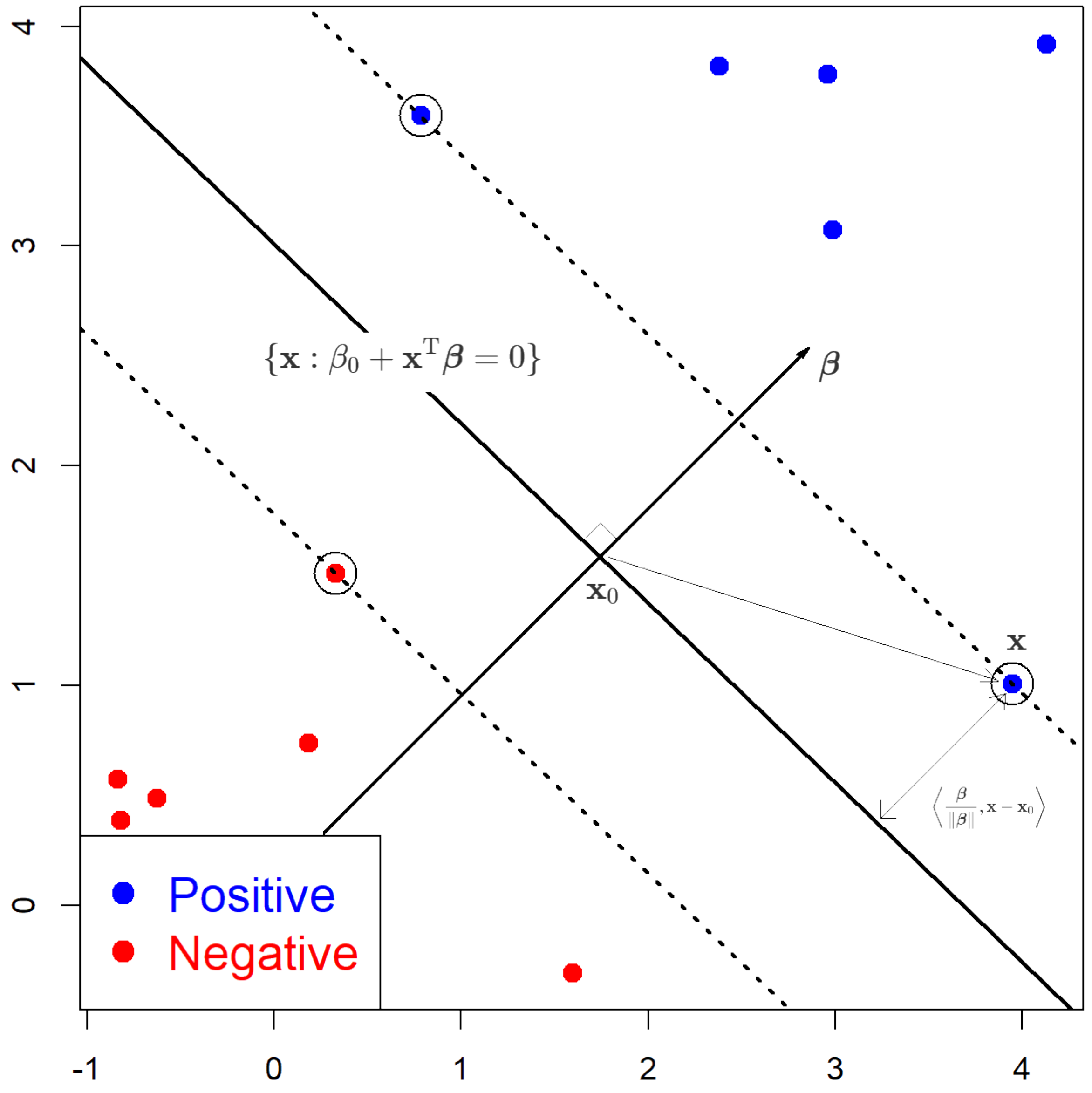
- Local linear regression
  - Local averaing: suffer bias at boundaries
  - Solution: local polynomial regression
- Multi-dimensional Kernels
  - Idea: utilize a certain distance measure, plug in to kernel function
  - Also suffer from the `curse of dimensionality` (similar to KNN), need to adjust h accordingly.

Week 7

- SVM
    - classification models
    - Maximum-margin Classifier
        - y_i : binary outcome from {-1, 1}
        - $$\hat{y} = \begin{cases} +1, & \text{if} \quad f(\mathbf{x}) > 0 \\ -1, & \text{if} \quad f(\mathbf{x}) < 0 \end{cases}$$
        - y_i * $f(x_i)$ > 0
        - best line: maximizes the margin
            - cost parameter C, with default value 1. has a significant impact on non-separable problems
            - separable case: need set to a very large value
            -



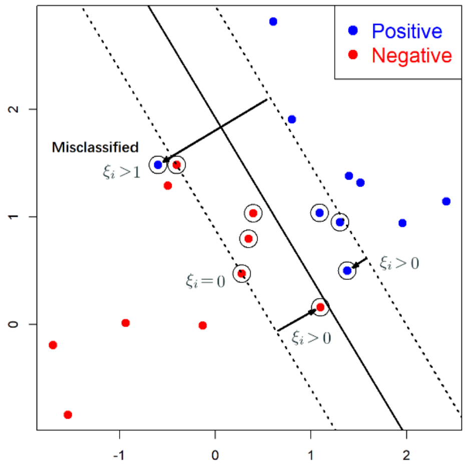
- Linearly separable SVM
  - Support vector: on the wrong side of the margin
  - The data set is on the correct side of the central line: correct separate. But, don't go beyond the margin line: call "slack variable". No separable case. Give a non-zero value to it. At least one to not be considered; everything less than that will not be considered.
  - Only the dots with a cycle around them are considered as a "support vector." (>=0) If > 0 rarely passes the central line.
  -



- Lineary Non-separable SVM with slack variables
  -

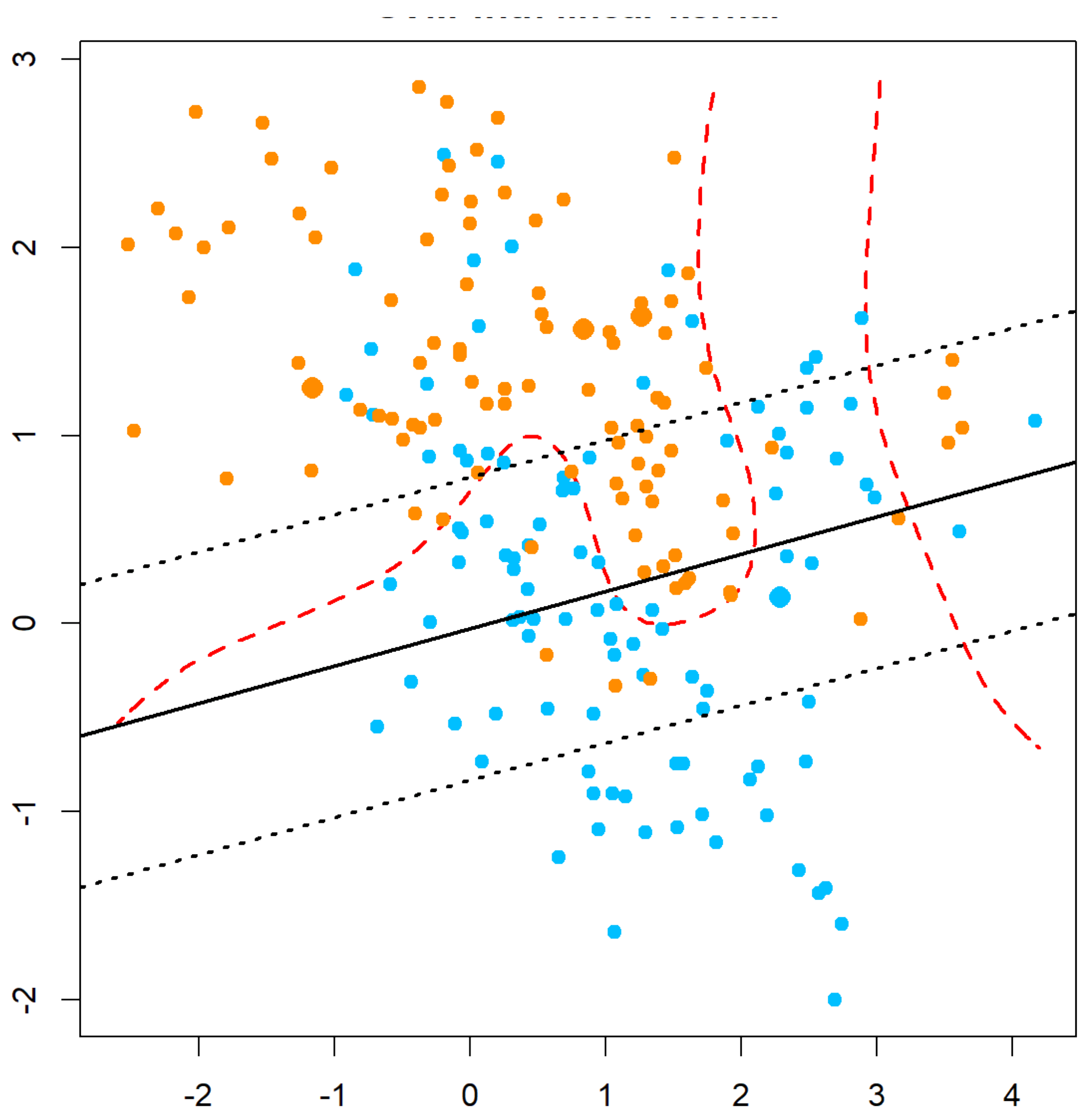
$$y_i(\mathbf{x}_i^T \boldsymbol{\beta} + \beta_0) \geq (1 - \xi_i)$$

for a positive ξ. Note that when $\xi = 0$, the observation is lying at the correct side, with enough margin. When $1 > \xi > 0$, the observation is lying at the correct side, but the margin is not sufficiently large. When $\xi > 1$, the observation is lying on the wrong side of the separation hyperplane.



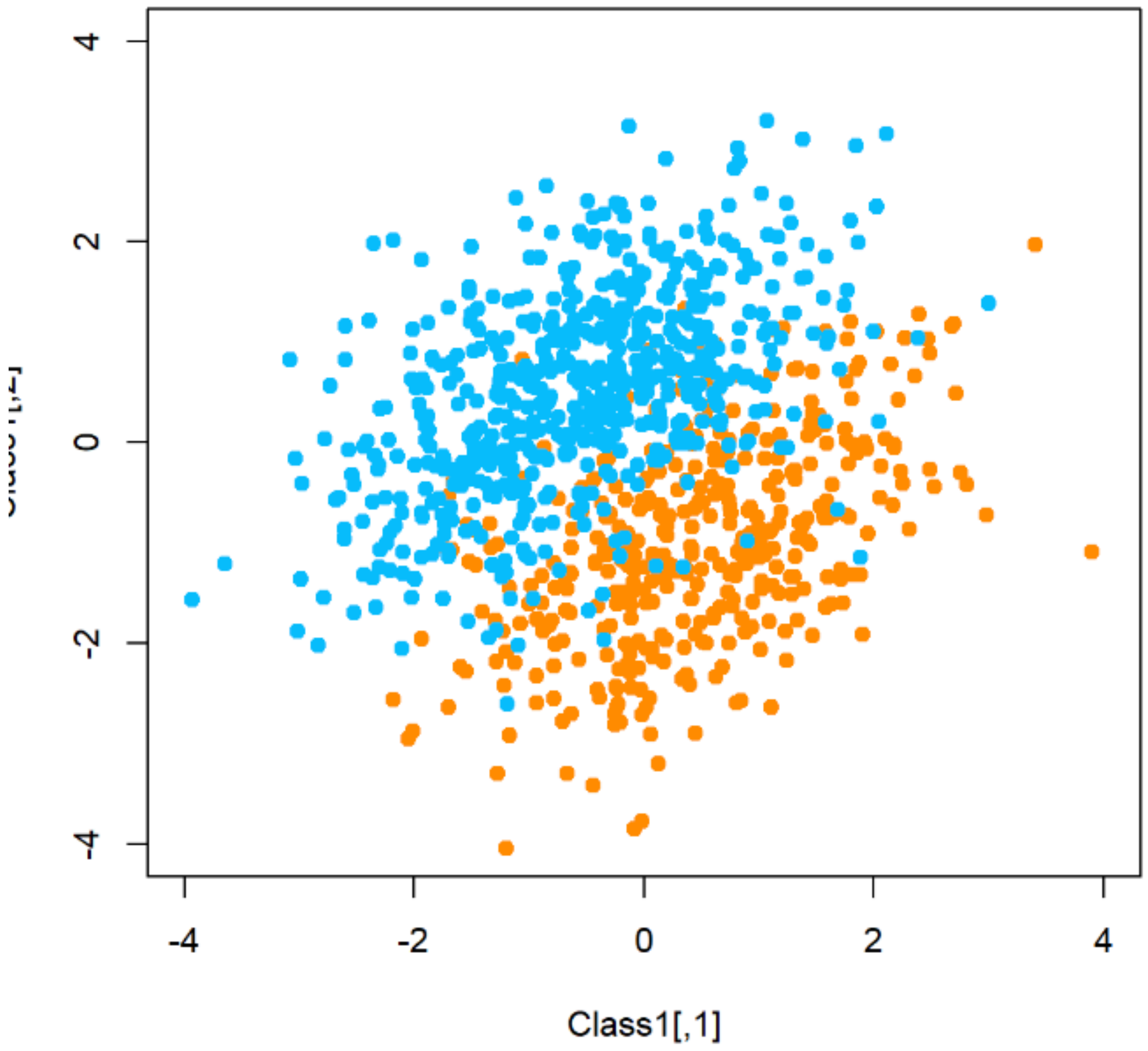
- Nonlinear SVM via Kernel Trick
  - Idea of kernel trick: using the kernel function of two obs x and z to replace the inner prduct between some feature mappin gof the two covariate vectors.
  - Advantages
    - Save computational cost
    -

SVM with linear kernal



Week 8

- Discriminant Analysis
  - Linear Discriminant Analysis(LDA) k个类别，线性决策边界
    -



- Estimating parameters in LDA
    -
        - Prior probabilities: $\widehat{\pi}_k = n_k/n = n^{-1} \sum_k \mathbf{1}\{y_i = k\}$, where n_k is the number of observations in class k.
        - Centroids: $\widehat{\boldsymbol{\mu}}_k = n_k^{-1} \sum_{i:\, y_i=k} x_i$
        - Pooled covariance matrix:

$$\widehat{\Sigma} = \frac{1}{n-K} \sum_{k=1}^{K} \sum_{i:\, y_i=k} (\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \widehat{\boldsymbol{\mu}}_k)^{\mathrm{T}}$$

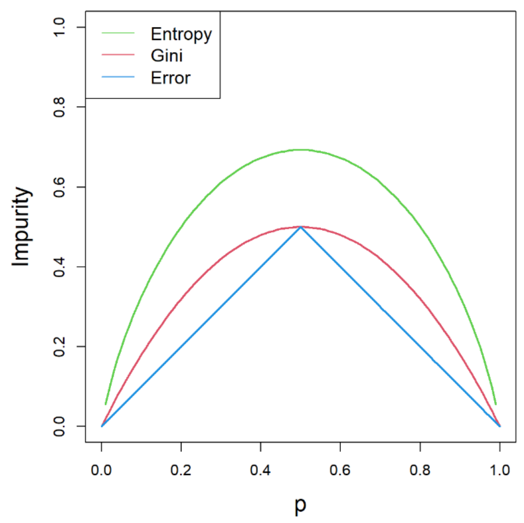
- QDA 非线性决策边界

- Bayes rule

$$f_B(\mathbf{x}) = \arg\min_f \ R(f) = \begin{cases} 1 & \text{if} \quad \eta(\mathbf{x}) \geq 1/2 \\ 0 & \text{if} \quad \eta(\mathbf{x}) < 1/2. \end{cases}$$

- Trees(can't use kernel trick)
  - Classification tree
    - A classification tree model is recursively splitting the feature space such that eventually each region is dominated by one class.
  - regression tree
    - trees are not ideal for regressions problems due to its large bias
    - Difference with classification tree: use a different way to evaluate how good a potential split is.
  - Splitting a node
    - compare all possible splits on all variables
      - At the current node, go through each variable to find the best cut-off point that splits the node.
      - Compare all the best cut-off points across all variable and choose the best one to split the current node and then iterate.
    - Criterion used to compare difference cut-off points
      - Gini impurity (CART)
      - Shannon entropy (C4.5)
      - Mis-classification error

The following plot shows all three quantities as a function of p, when there are only two classes, i.e., K = 2.



- - Smaller value--node is more "pure", higher certainty when predicting a new value
- The whole idea of splitting nodes: want the two resulting child nodes to contain less variation, and want the child node to be as "pure" as possible. (So calculate this error criterion both before and after the split and see what cut-off points gives us the best reduction of error)
- ----later not analyzed

- Tuning
  - when to stop splitting
  - model is not smooth. We will later learn the random forest model that try to address this issue.
- M-try smaller, tree more random. Then, if you want a more stable model, you may need a larger tree. That is talking about every single tree. The computation cost is the computation for each m-try.

Week 9

- Random Forests (can't use kernel trick)
  - parallelly fitted many CART models with some randomness
    - Bootstrapping of data for each tree using the **Bagging idea**, and use the **averaged** result (for regression) or **majority voting** (for classification) of all trees as the prediction.
    - At **each internal node**, we may not consider all variables. Instead, we consider a **randomly selected `mtry` variables** to search for the best split.
    - For **each tree**, we will not perform pruning. Instead, we simply stop when the internal node contains no more than `nodesize` **number of observations**.
  - Bagging Predictors
    - different decision rule
      - CART: decision line has to be aligned to axis.(difficult when dealing with non-axis-aligned decision boundaries)

decision boundaries)

        - Bagging: (fit many CART model, decision bound is more smooth)

    - PS. "True" decision rule = Bayes rule

- Random Foresots

    - `ntree`: number of trees
    - `sampsize`: how many samples to use when fitting each tree
    - `mtry`: number of randomly sampled variable to consider at each internal node
    - `nodesize`: stop splitting when the node sample size is no larger than `nodesize`

- Effect of `mtry` and `nodesize` -- Tradeoff

    - mtry

        - **small mtry**: by chance randomly select some irrelevant variables that has nothing to do with the outcome. Then this particular split would be wasted. Missing the true variable may cause **larger bias**.
        - Large: greedy for signals since we compare many different variables and pick the best one. Risk of **over-fitting**.

    - nodesize

        - small nodesize: over-fitting
        - Large nodesize: under-fitting

- Categorical predictors

    - RF can direcly handle categorical predictors without coding into dummy variables

- Kernel function: counts how many times x falls into the same terminal node as observation x_i

- Adaptiveness of RF kernel

    - adapt well in **high-dim**, sparse setting
    - because of greedy splitting rule selection, ignore covariates not effective on explaining the variation of Y
    - Hence, make model similar to a kernel method on low-dim space. Outcome is only related to the first dim.
    - tuning parameter `mtry`

        - Large: very greddy on selecting the rule signal variable to split. In high dim, only use few variables before reaching the terminal node, making the model only rely a few dimensions.
        - Small: behave simiarly to a regular kernel estimator with good smoothing(small var). But since randomly selecting a dim to split, the bandwidth on each dim would also be similar but large.

- Correlation of Tree Predictions

    - nodesize and mtry are both tuning parameters

        - **Out-of-bag**(OOB) prediction error: build-in CV method in random forests. Out-of-bag represents the data pts not used in constructing a particular tree. Often used to select tuning parameters.
            - Train all trees, keep track the OOB sampels while training each tree
            - For each obs, all trees that do not include this sample is training data. Treat as a smaller forest and gain its predicted value.
            - For each obs, gain OOB prediction and average their errors.
    - Variable Importance
        - identify which features (or variables) are most influential in predicting the outcome based on the fitted model.

    - lots of tuning parameters, a combination of them makes some sense
    - Kernel: if fit RF, this is a way to fit your model in a kernel format(same to the kernel concept in the Trees, KNN), which is difference the the kernel in the SVM.

- Boosting
    - reduce the randomness of random forests

Week 10 Unsupervised Learning Algorithms

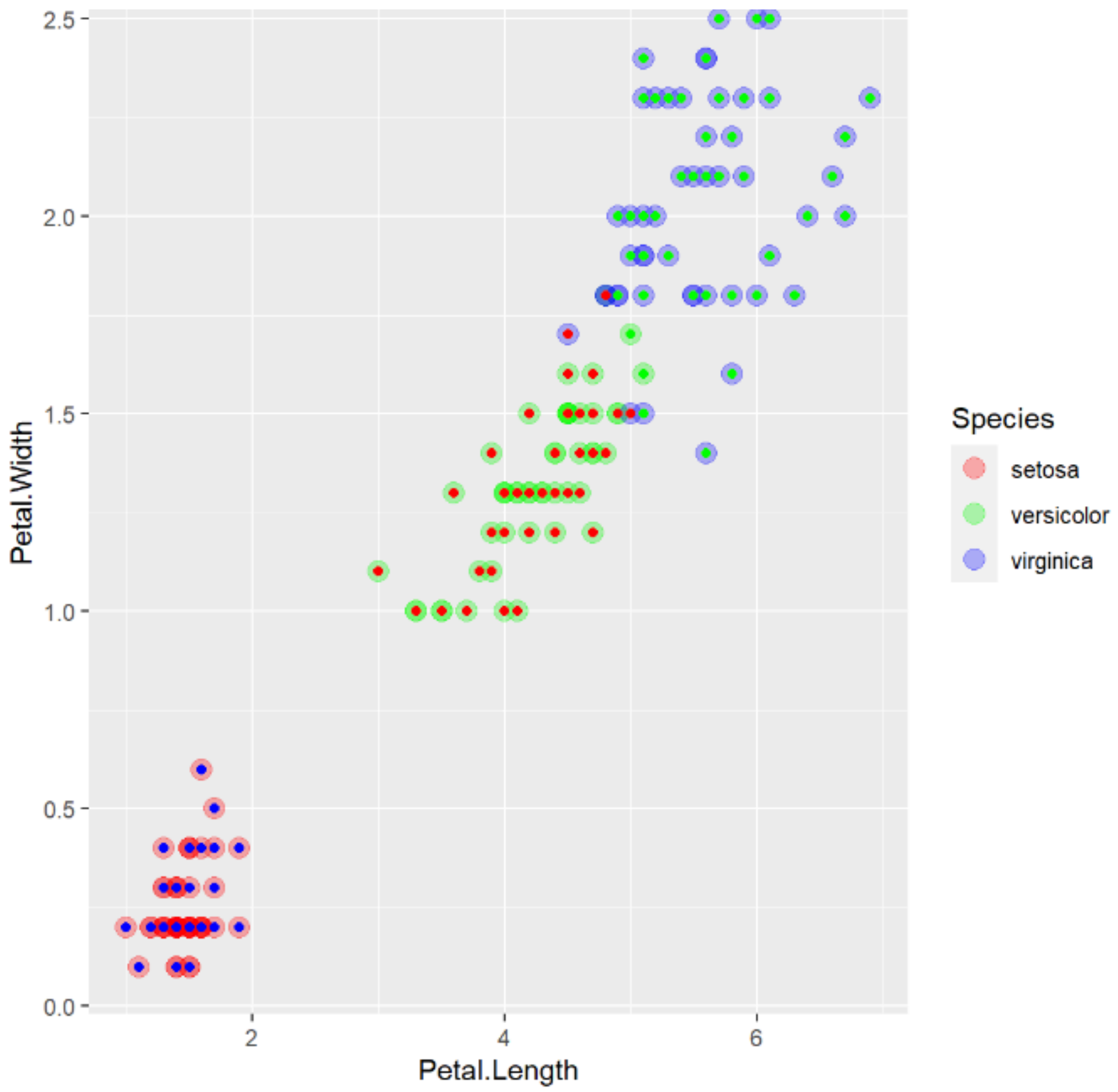
- PCA
    - SDK View
        - X = UDV^T
    - Optimization View
        - Minimize $||$ X - $ud \ v^T$ $||_2^2$
    - Centering Issue
        - always use `scale()` to center the variables at beginning, but only center, not scale
        - If not center, first PC could be a direction that points to the center of the data
        - Whether using scaling depeneds on the particular application
            - Use scaling: extremely disproportionate variables, eg. Age vs. RNA expression
            - Not use scaling: all variables are of the similar types, eg. color intensities of pixels. (variables with larger variations may carry more signals, scaling more lose that information.)

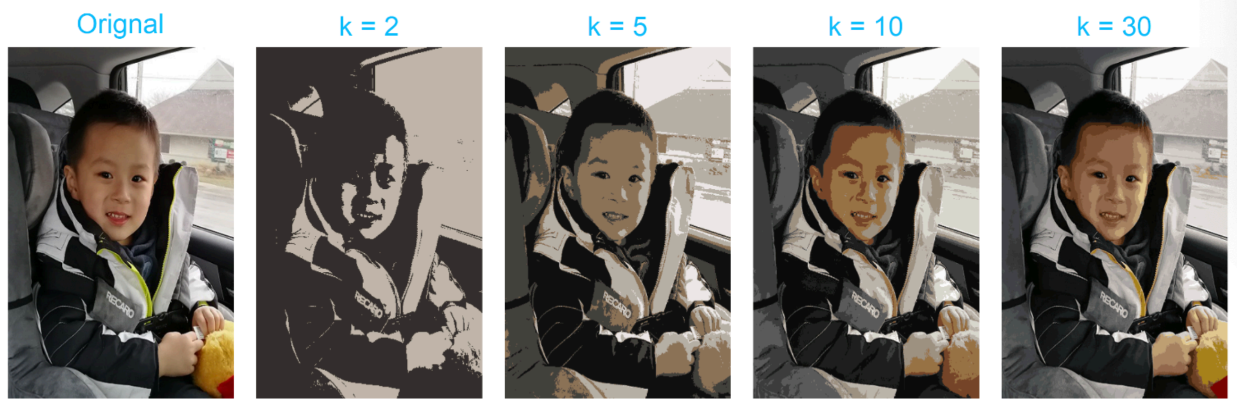
- K-Means(difference with map: have a relationship with each other or not?)
    - K-means clustering algorithm attemps to solve the optimization problem:

$$\min_{C, \{m_k\}_{k=1}^K} \sum_{k=1} \sum_{C(i)=k} \|x_i - m_k\|^2,$$

where $C(\cdot) : \{1, \dots, n\} \to \{1, \dots, K\}$ is a cluster assignment function, and m_k's are the cluster means. To solve this problem, k-means uses an iterative approach that updates $C(\cdot)$ and m_k's alternatively. Suppose we have a set of six observations.

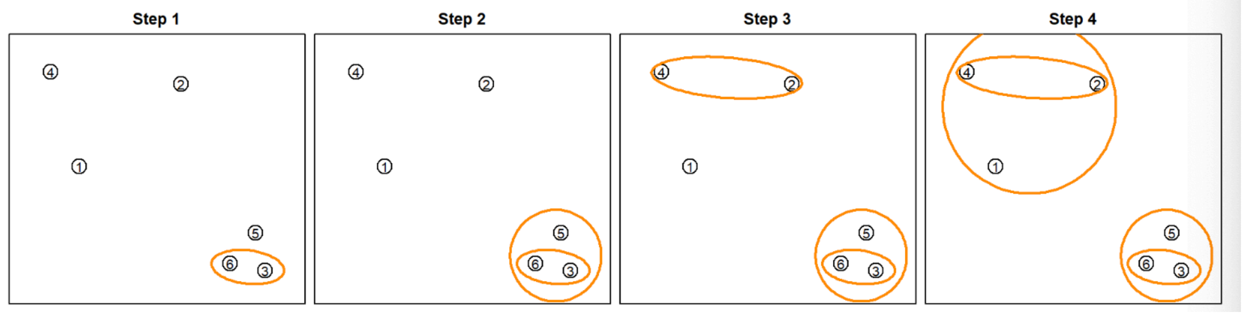
-
    - First randomly assign them into two clusters
    - calculate the corrsponding cluster mean
    - assign each obs to closest cluster mean
    - Recalculate the cluster mean, repeat until nothing to move
- start with a *ramdom* one and the obj func is not convex, may obtain different result with different start values
-

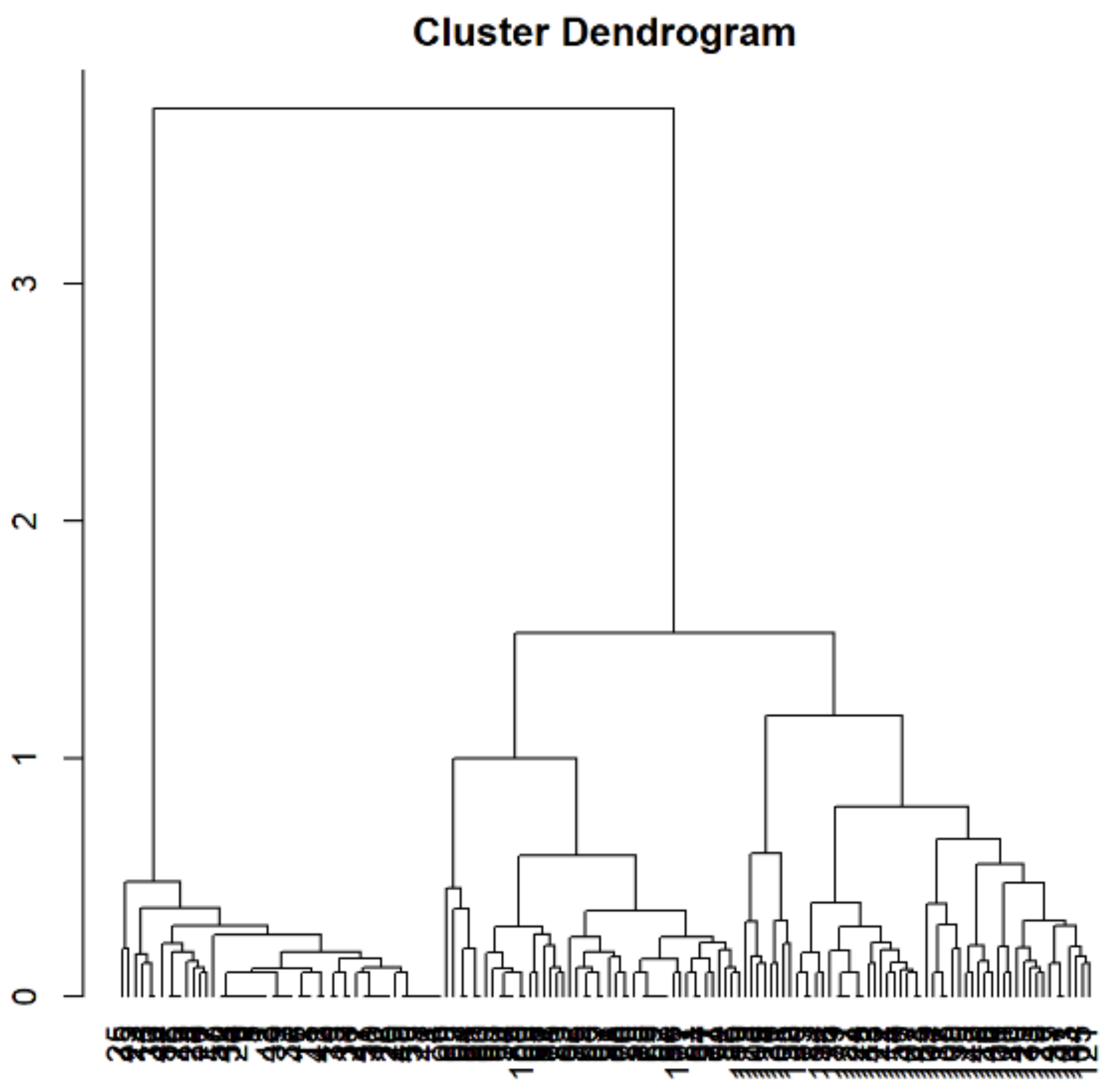


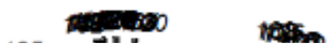
- 

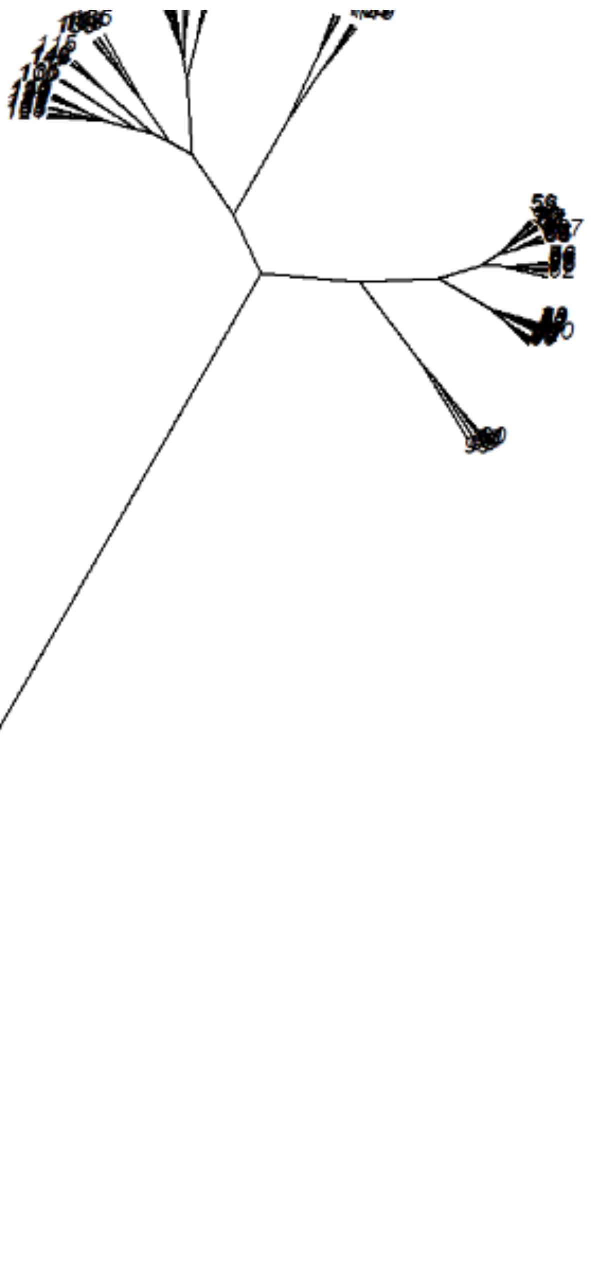
- Hierarchical Clustering
  - The goal is to progressively group them together until there is only one group. During this process, we will always choose the closest two groups (some may be individuals) to merge.

process, we will always choose the closest two groups (some may be individuals) to merge.

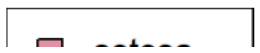


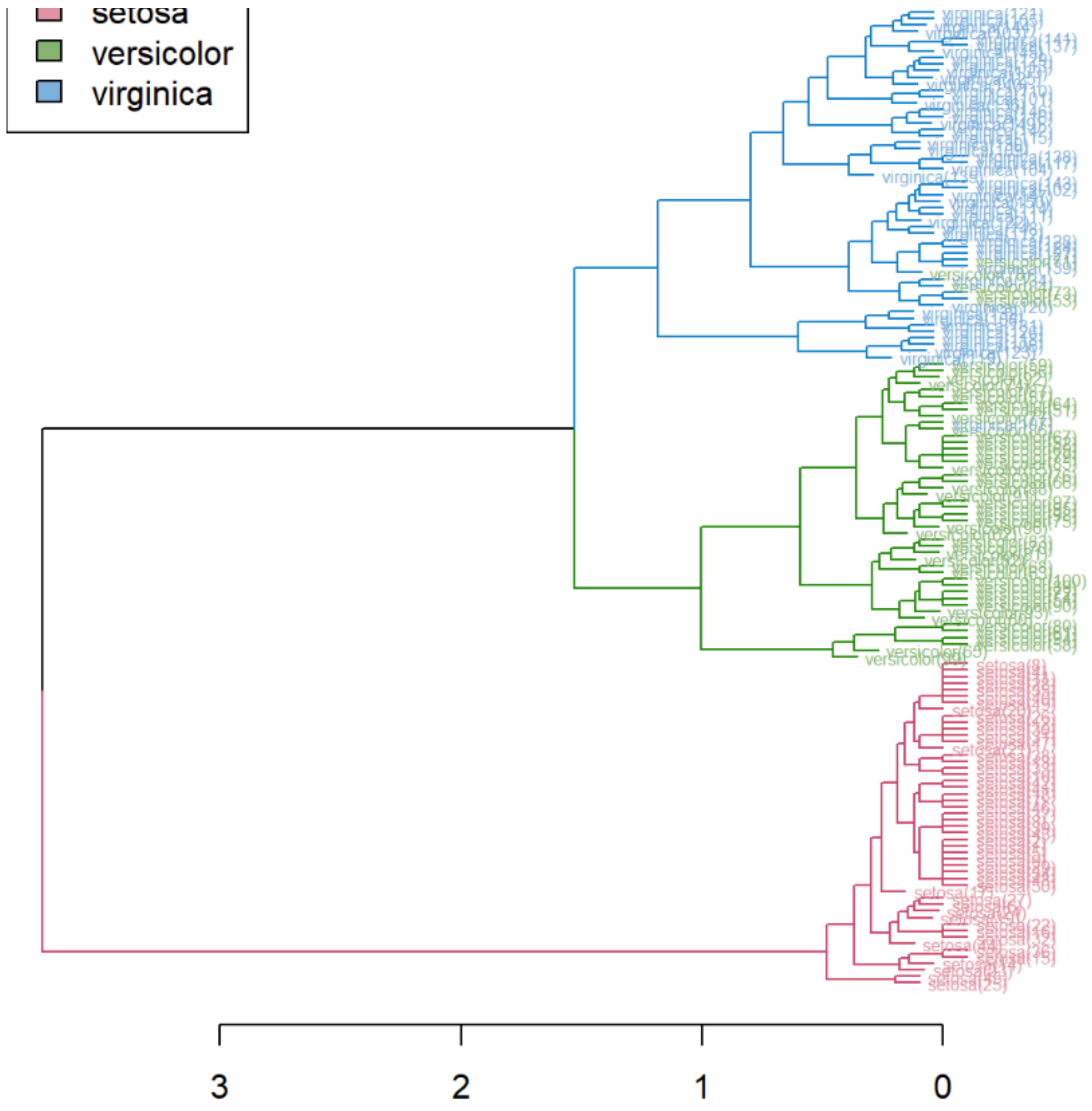
- Height of each split: how separated the two subsets are(the distance when tehy are merged)
- Select the number of clusters: pick a cutooff where the hieght of the next split is short
    - 

    - 

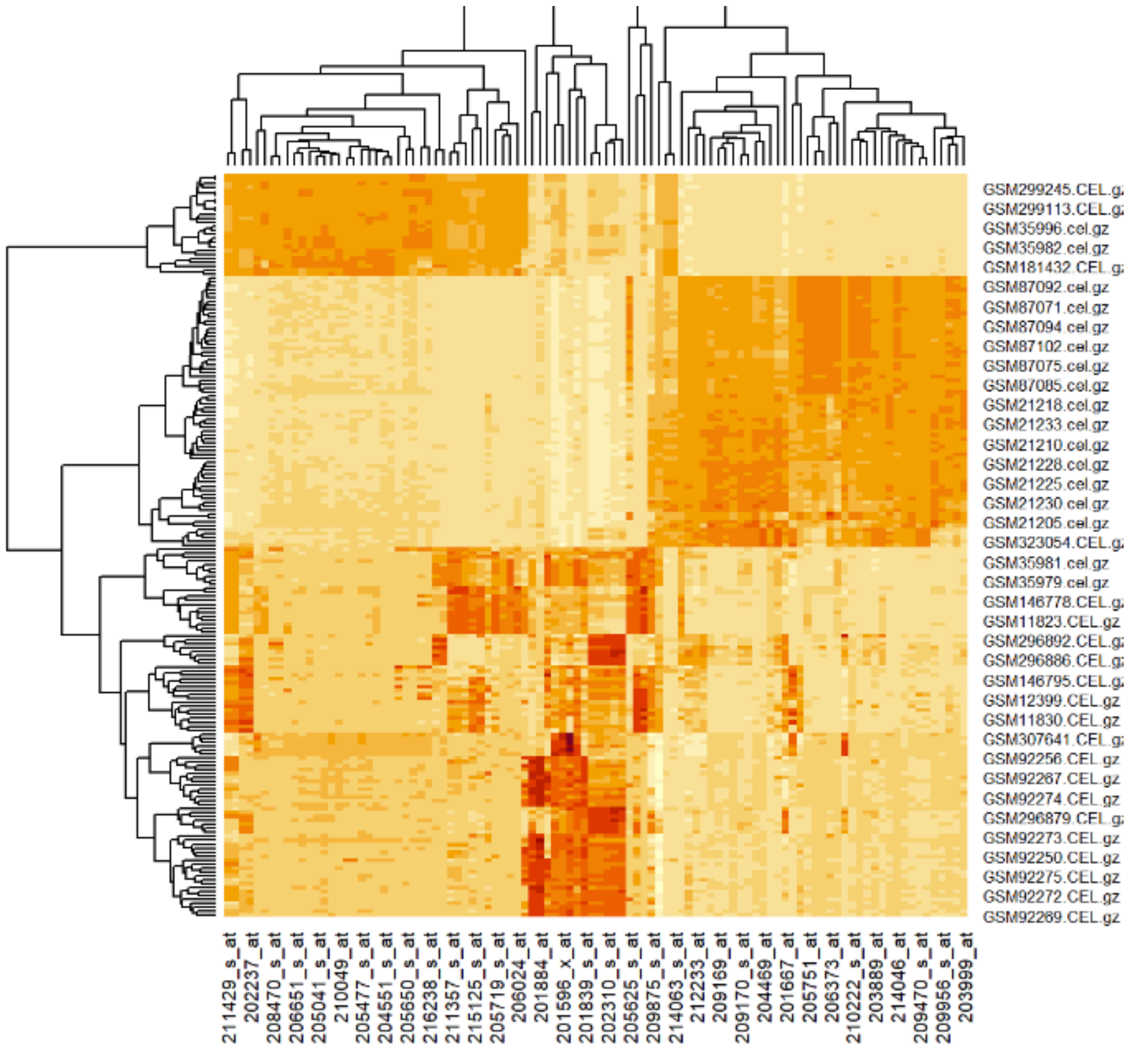

25
33

▪



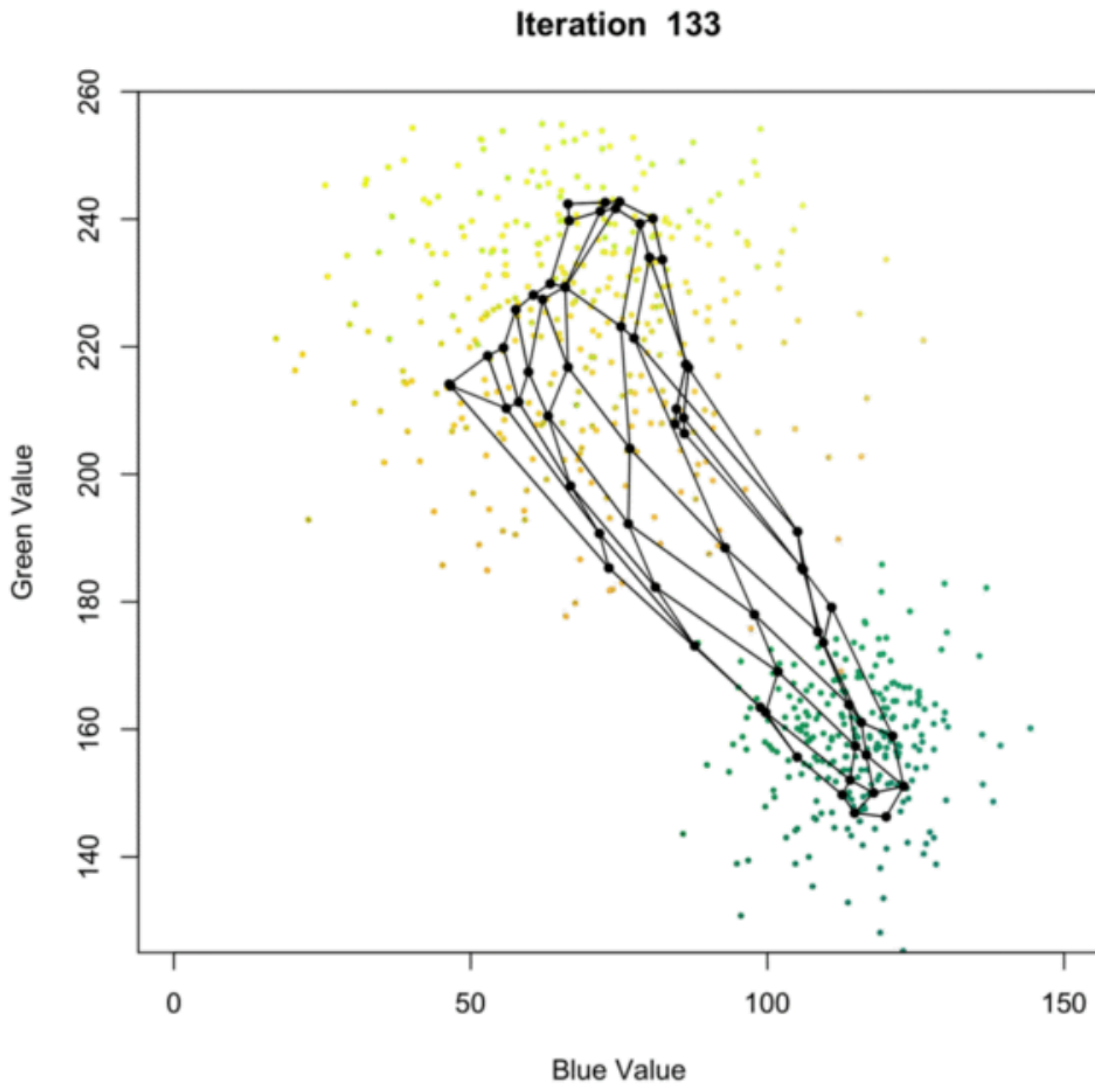

setosa
versicolor
virginica
3
2
1
0

■



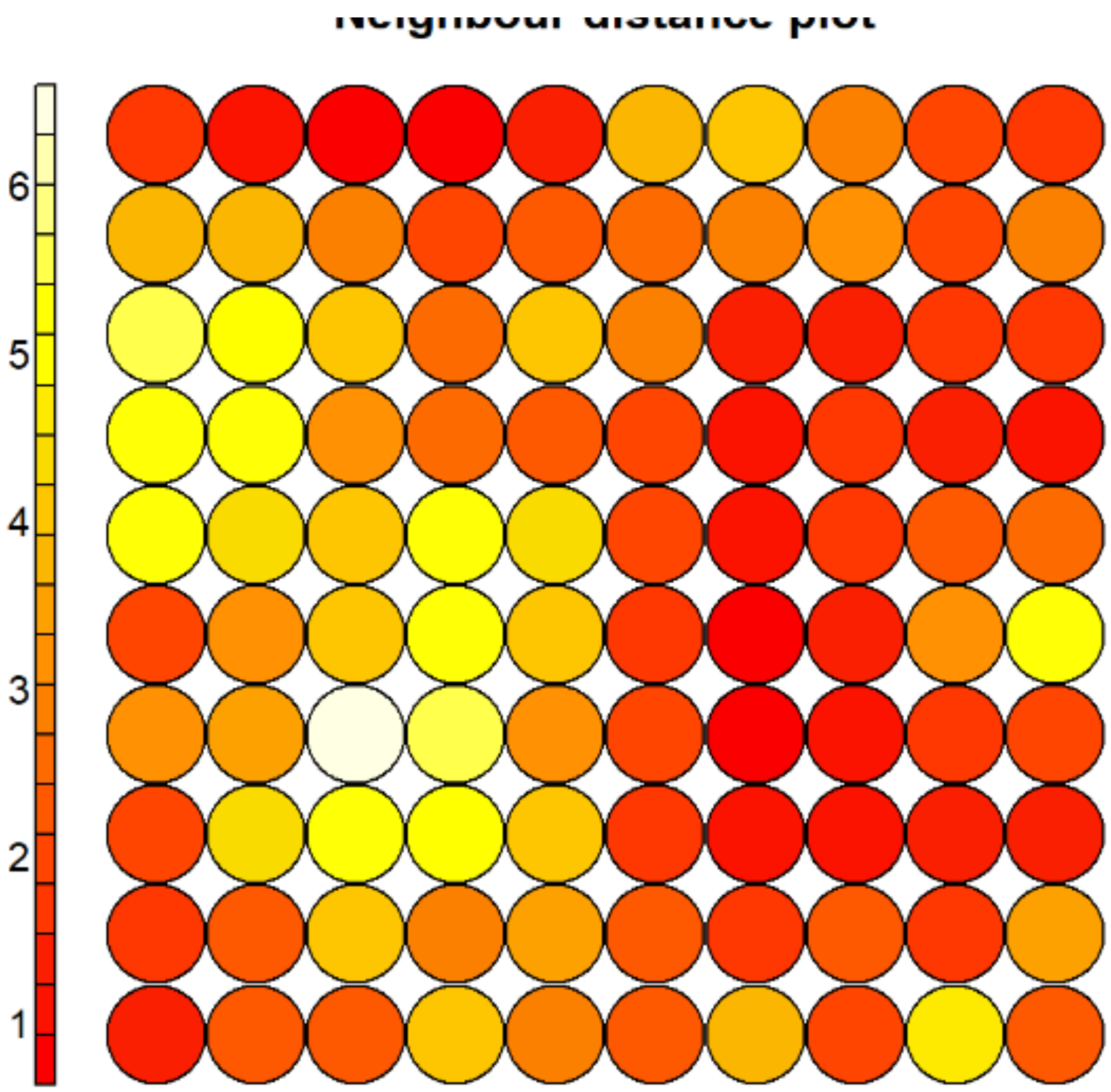
- Spectral Clustering
  - Cluster obs based on their proximity information
    - Two steps:
      - feature embedding/demension reduction(construct the normalized version-represent the proximity information, then perform eigen-decomposition of the matrix)
      - Regular clustering algorithm, eg. k-means
  - Adjacency Matrix W
    - use nonlinear way to describe the distance between subjects
  - Laplacian Matrix(several choices)
    - L = D - W, D: diagonal matrix with its elements equation to the row sums of W
    - then perform eigen-decomposition on $\mathbf{L}$L to extract/define some underlying features.
  - Feature Embedding嵌入
    - smallest eigenvalue: always 0
    - use eigen vectors associated with the second and third smallest eigen values, which define

- use eigen-vectors associated with the second and third smallest eigen-values, which define some feature embedding or dimension reduction to represent the original data
- Since we know the underlying model, two dimensions are enouth
- Then perform k-means on these two new features

- Self-organizing Map
  - 

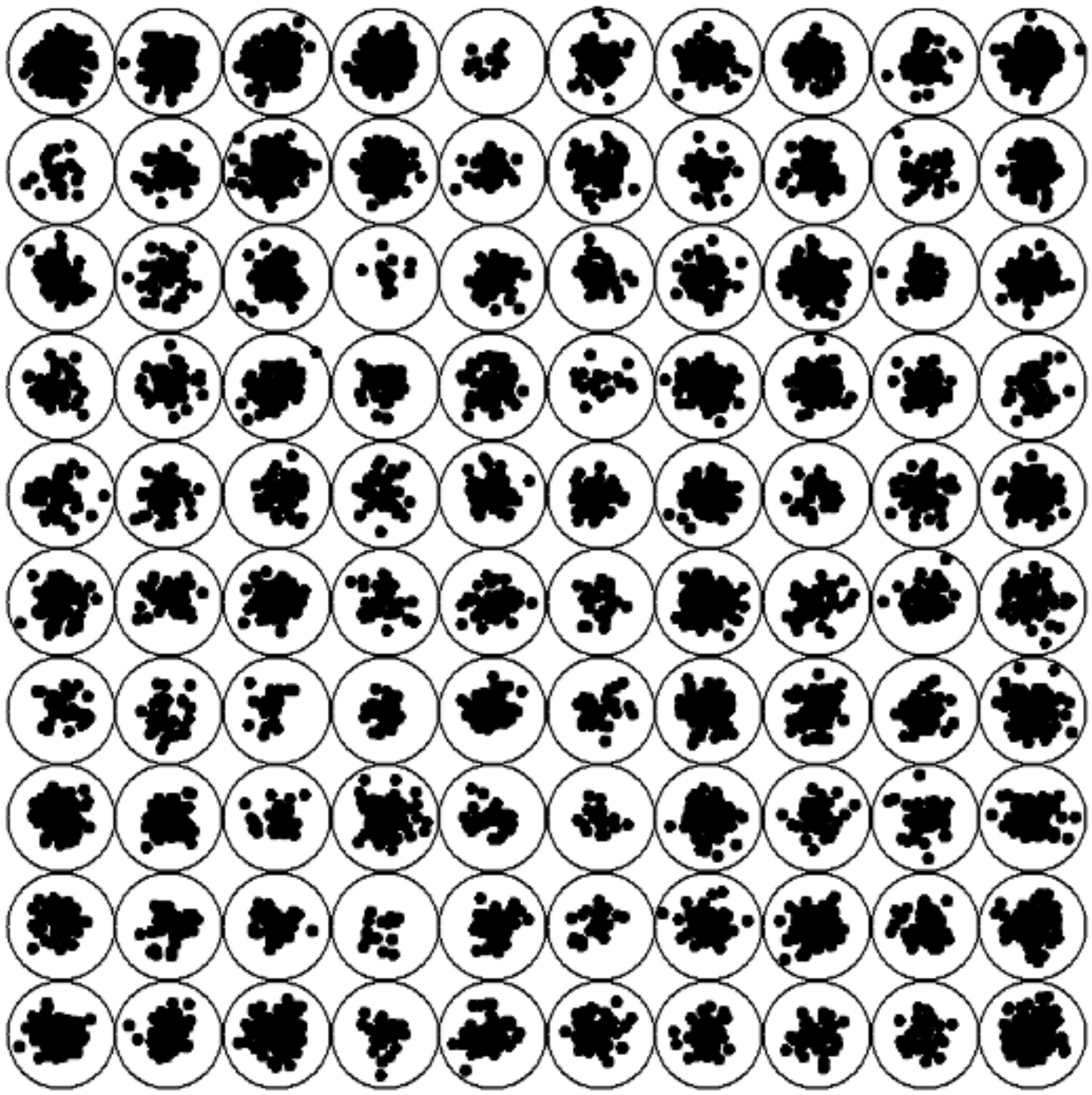
  -

Neighbour distance plot

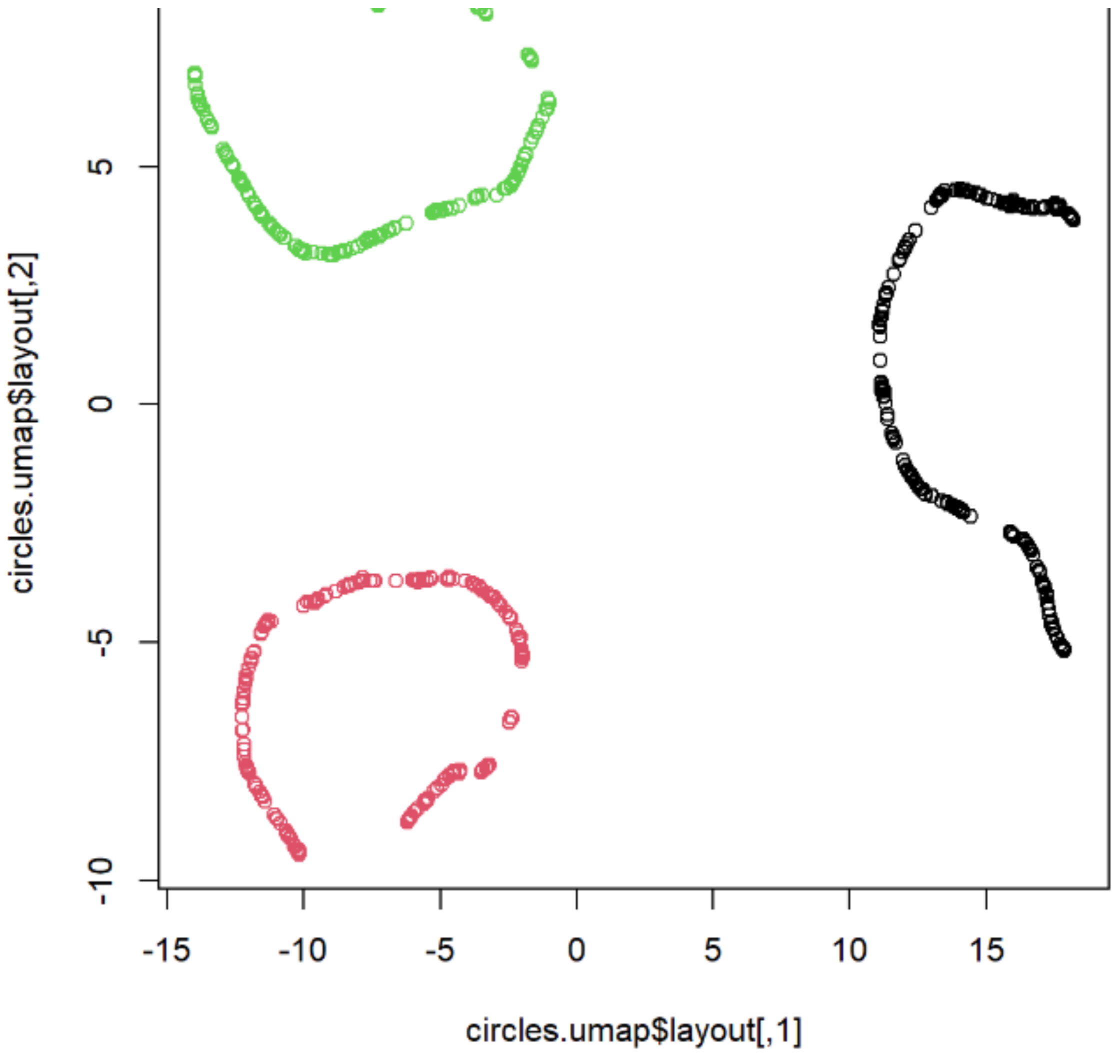


-

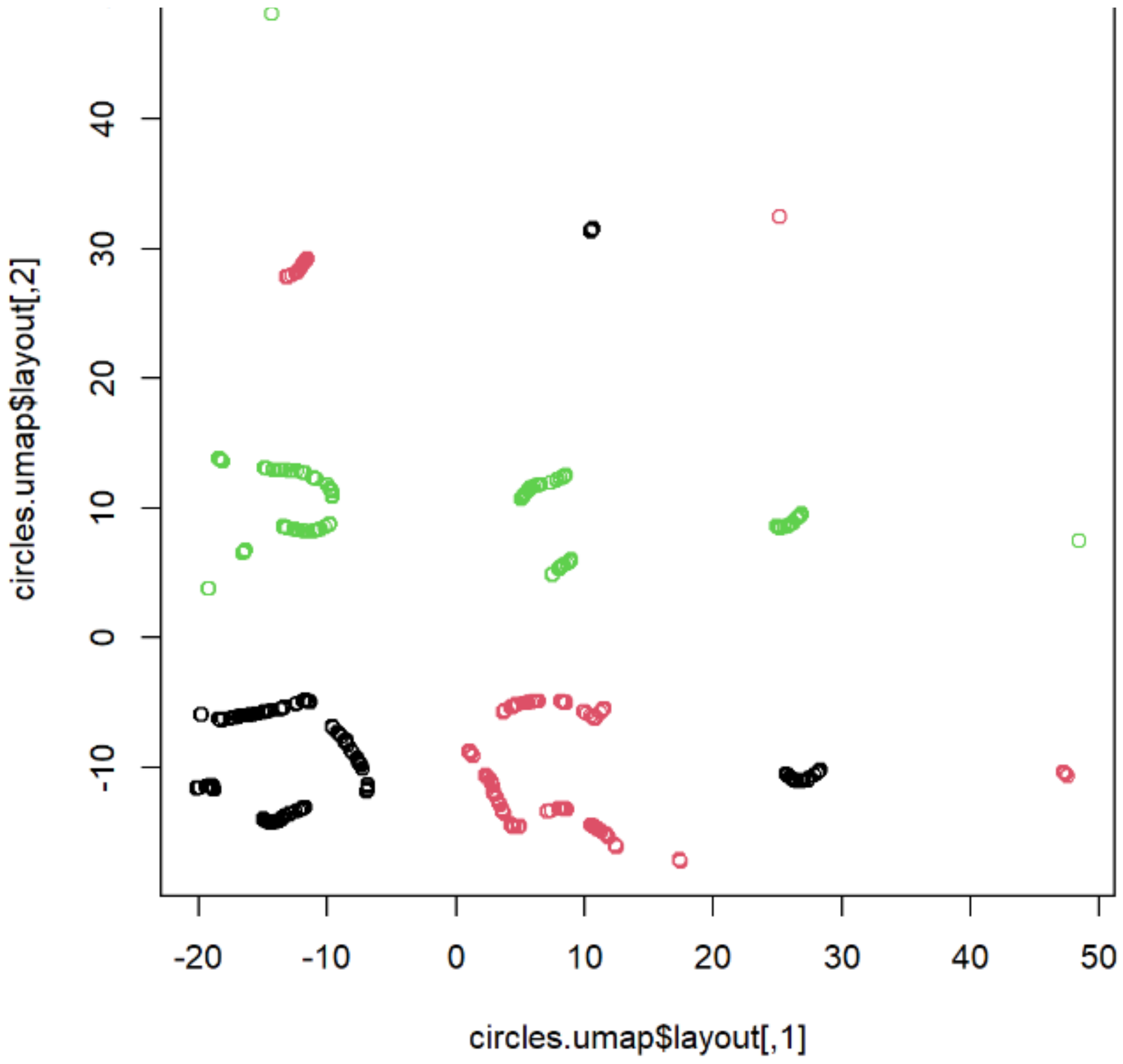
Mapping Type SOM



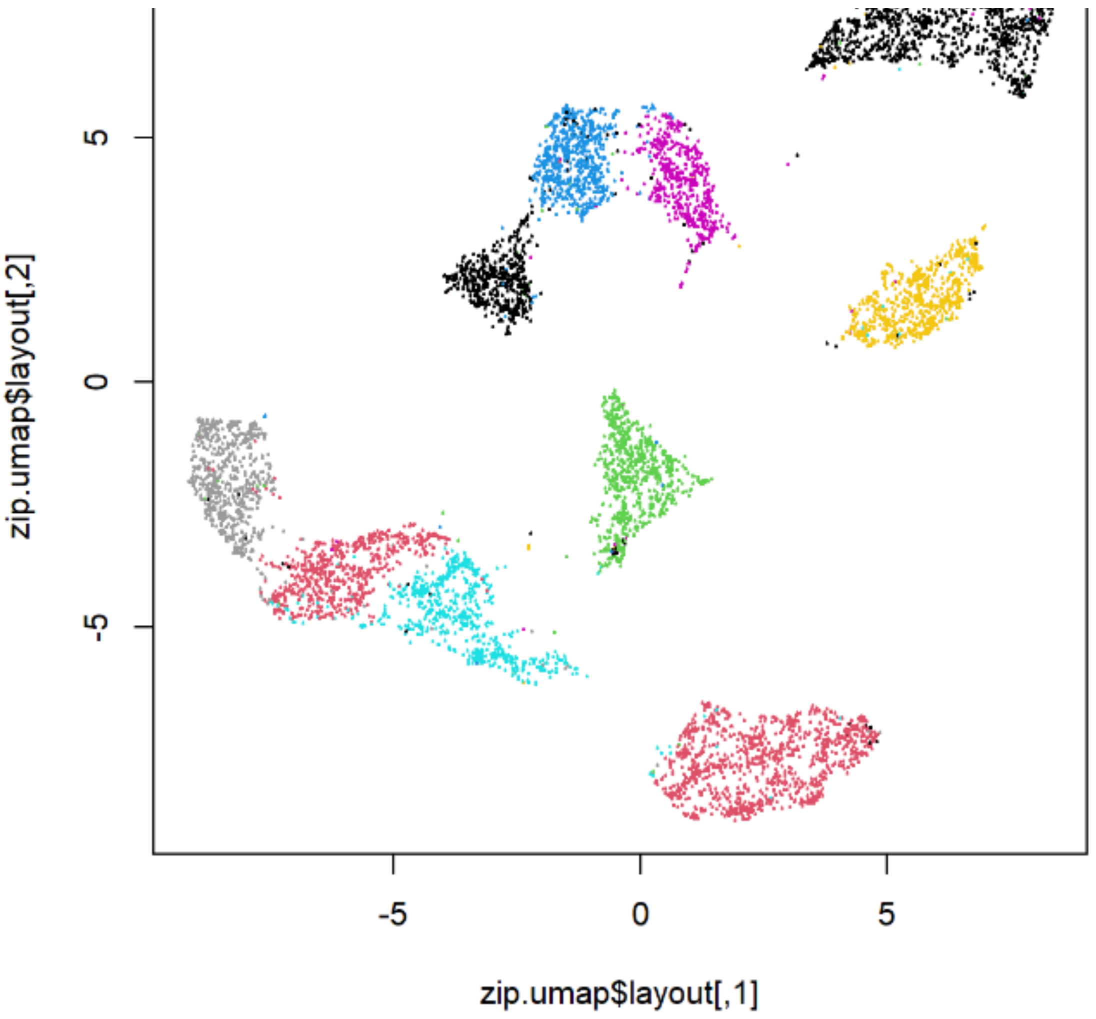
- Uniform Manifold Approximation and Projection(UMAP)
    - feature embedding/dimension reduction algorithm
    - Two other methoes:
        - t-SNE, graph values are proportional to the kernel density function between two points with a t-districution density function.[based on the Kullback-Leibler divergence]
        - spectral embedding.[eigen-values, meaning the matrix approximation]
    - Main difference: [how they define "similar"]
    - UMAP: [a type of cross-entropy]
        - If use default tuning, will create a two-dim embedding
        - We can see that UMAP learns these new features, which groups similar observations together.
        -



- Tuning
  - lots of tuning parameters
  - Key: how to create the KNN graph in the first step
  -



- Usually we don't use values larger than three, so the result is not as perfect as we wanted. It seems that there are more groups, although each group only involves one type of data. There are other parameter you may consider tuning.
- Another example on larger data
    -



zip.umap$layout[,2]
zip.umap$layout[,1]
5
0
-5
-5
0
5