Stat 115 Lab 4

PCA, SVM, BWA

(your name)

February 13-14, 2018

Roadmap

- Dimension reduction: summarizing complicated data.
- Classification: Predicting $y \in \{0, 1\}$ from $\mathbf{X}$.
- Read mapping: finding regions of the genome where short reads map to.

Install and Load Packages

```
# install packages from bioconductor
source("https://bioconductor.org/biocLite.R")
biocLite("sva")
biocLite("bladderbatch") # for the example data
install.packages("class")
install.packages("e1071")
install.packages("ggplot2")
install.packages("cowplot")
install.packages("caret")
# etc.
```

```
library(sva)
library(bladderbatch)
library(limma)
library(ggplot2)
library(cowplot)
library(class)
library(e1071)
library(caret)
```

Load Data

- Gene expression data from investigation into bladder cancer.
- Outcome: finding differentially expressed genes that are associated with cancer status (0/1 in the variable `hasCancer`).
- Already normalized with RMA.

```
data(bladderdata)
pheno <- pData(bladderEset)
pheno$hasCancer <- as.numeric(pheno$cancer == "Cancer")
edata <- exprs(bladderEset)
head(pheno)
```

```
##              sample outcome batch cancer hasCancer
## GSM71019.CEL      1  Normal     3 Normal         0
## GSM71020.CEL      2  Normal     2 Normal         0
```

```
## GSM71021.CEL      3  Normal     2 Normal         0
## GSM71022.CEL      4  Normal     3 Normal         0
## GSM71023.CEL      5  Normal     3 Normal         0
## GSM71024.CEL      6  Normal     3 Normal         0
```

PCA

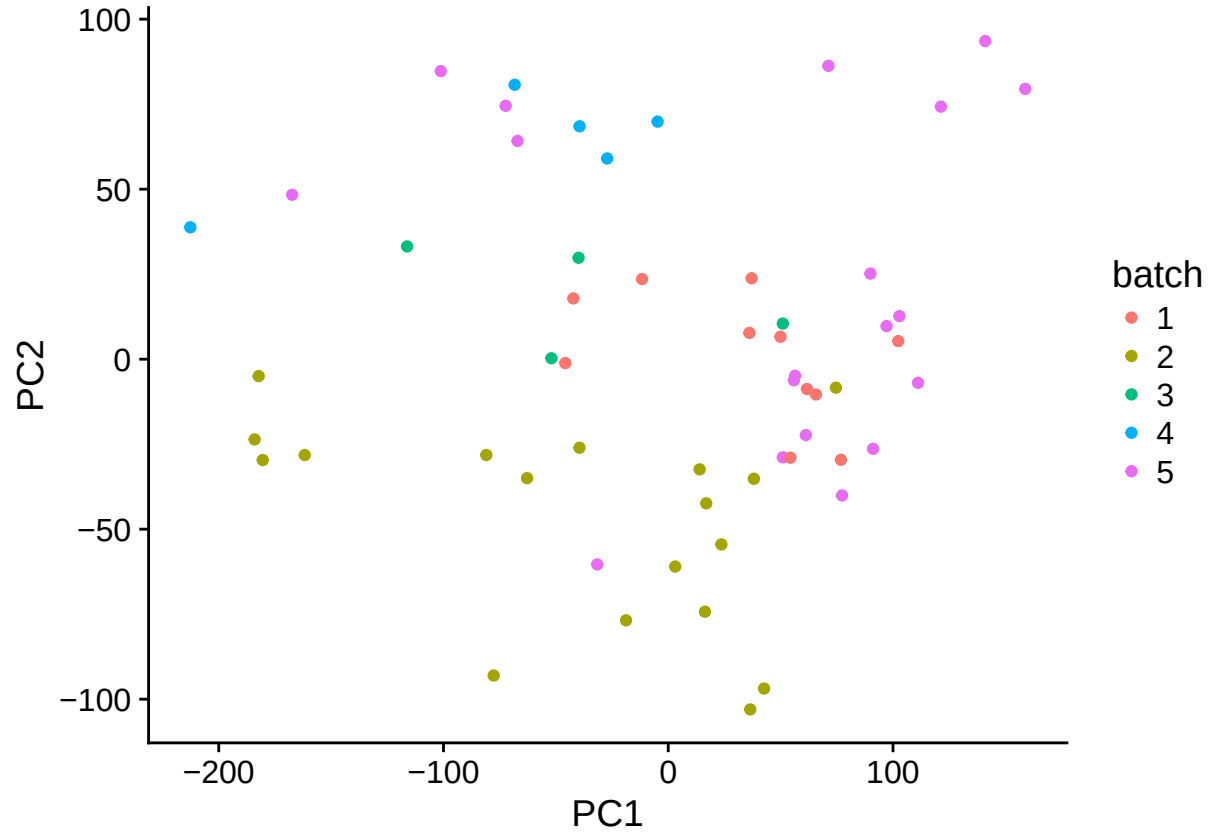
- Finds the best linear combinations of the variables.
- "Best" means optimally describing the variance.
- Can produce lower-dimensional summaries of the data.
- Useful for visualization, among other things.

PCA

- Main function: `prcomp`.
- Definitely want to center and scale your data: e.g. for car data, you might have 4-8 cylinders, but weight could be measured in kilograms or grams.

```
pca_raw <- prcomp(t(edata), center = TRUE, scale. = TRUE)
edata_pc_df <- as.data.frame(pca_raw$x)
edata_pc_df$batch <- as.factor(pheno$batch)
edata_pc_df$hasCancer <- as.factor(pheno$hasCancer)
#head(edata_pc_df)

ggplot(edata_pc_df, aes(x = PC1, y = PC2, color = batch)) +
    geom_point()
```



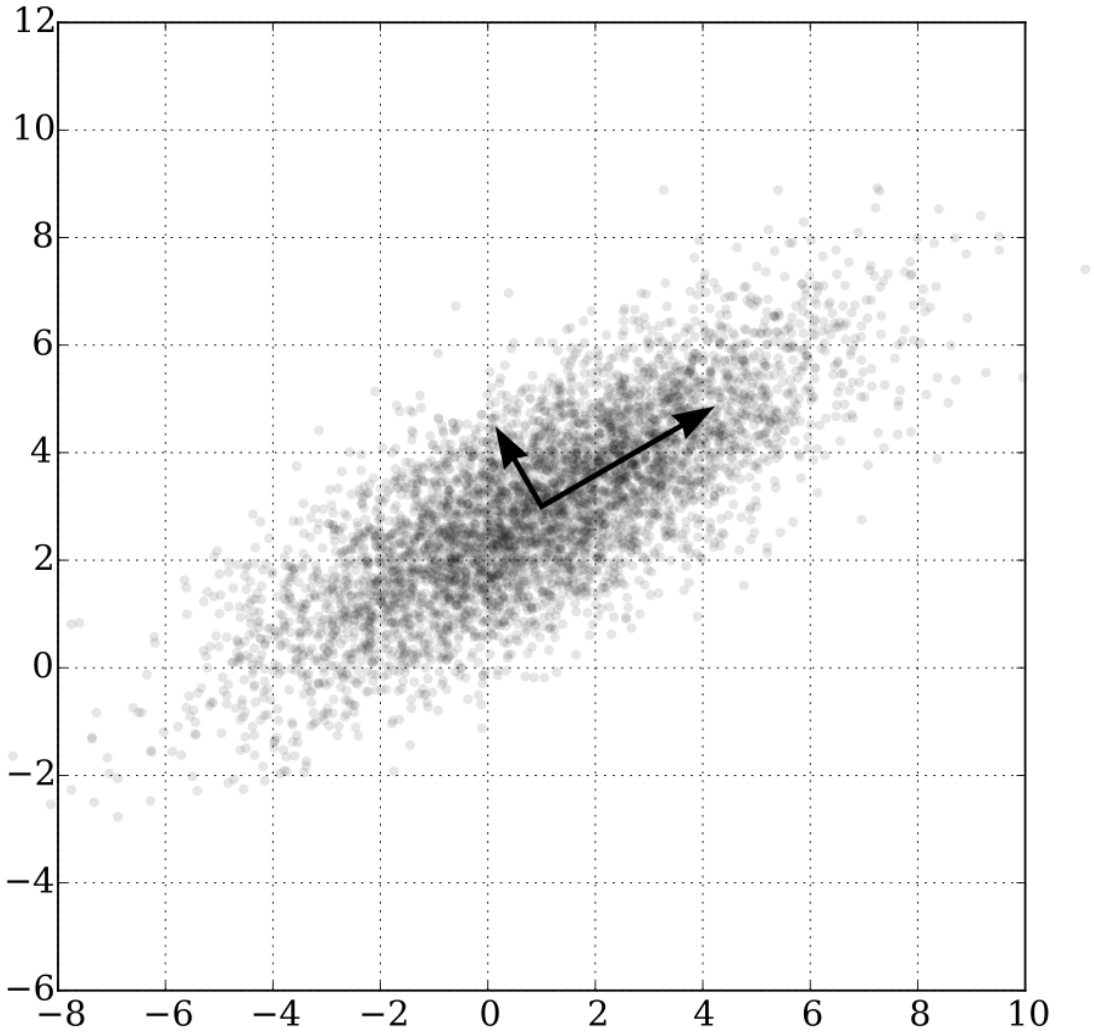
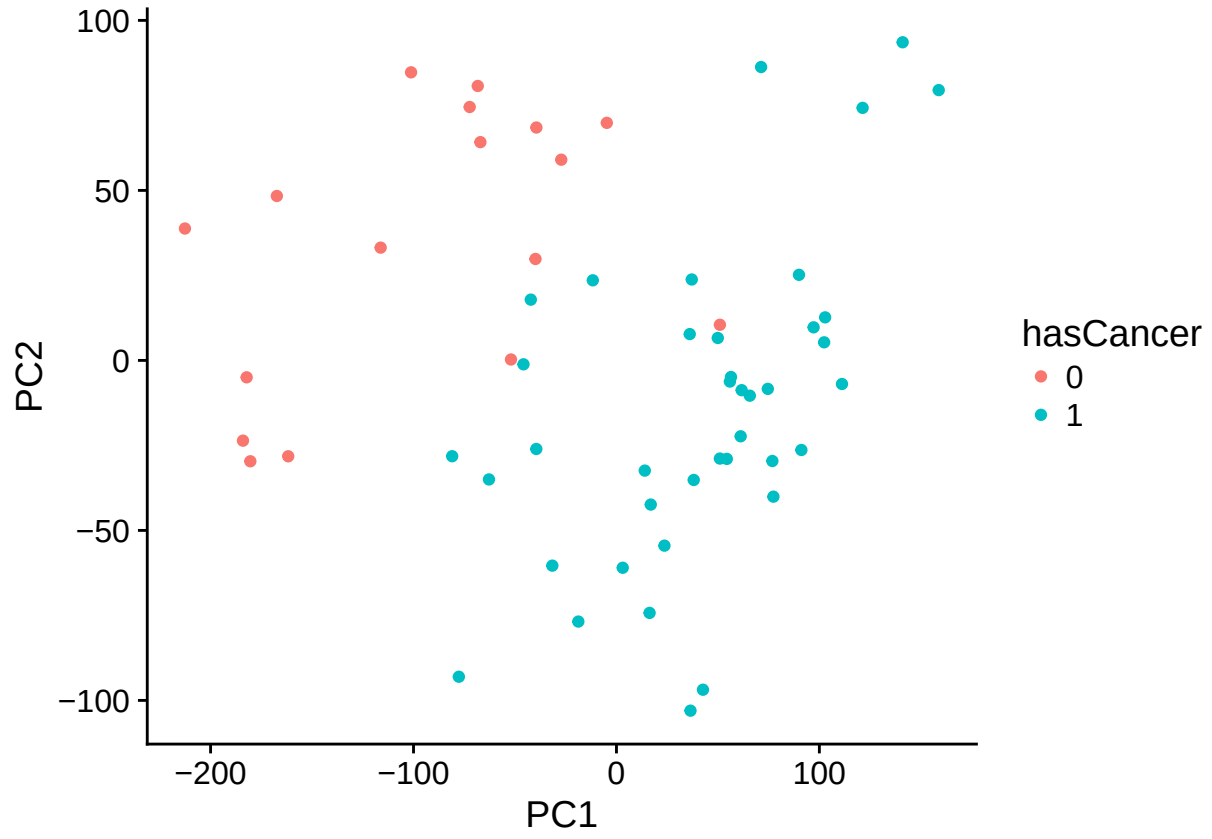


Figure 1: Source: https://en.wikipedia.org/wiki/File:GaussianScatterPCA.svg

```
ggplot(edata_pc_df, aes(x = PC1, y = PC2, color = hasCancer)) +
    geom_point()
```



PCA Variance Explained

- Linear algebra result: $\text{trace}(\Sigma) = \sum_i \lambda_i$
- $\text{trace}(\Sigma)$ can be thought of as total variance.
- Variance of PC_i is λ_i
- So variance explained by PCs 1 to k is $\frac{\sum_{i=1}^{k} \lambda_i}{\sum_{i=1}^{K} \lambda_i}$
- Denominator is sum of all eigenvalues
- Given the formula here, can you write code to plot the variance explained from 1 to k, for all possible values of k $(1, 2, \ldots, 57)$?

PCA Variance Explained: Your turn

```
# your turn
```

PCA After ComBat: Does it change?

```
# your turn
# run combat on edata to remove batch effect (see Lab 3)
```

```r
# your turn
# run PCA on the data from ComBat, draw a plot of the result
```

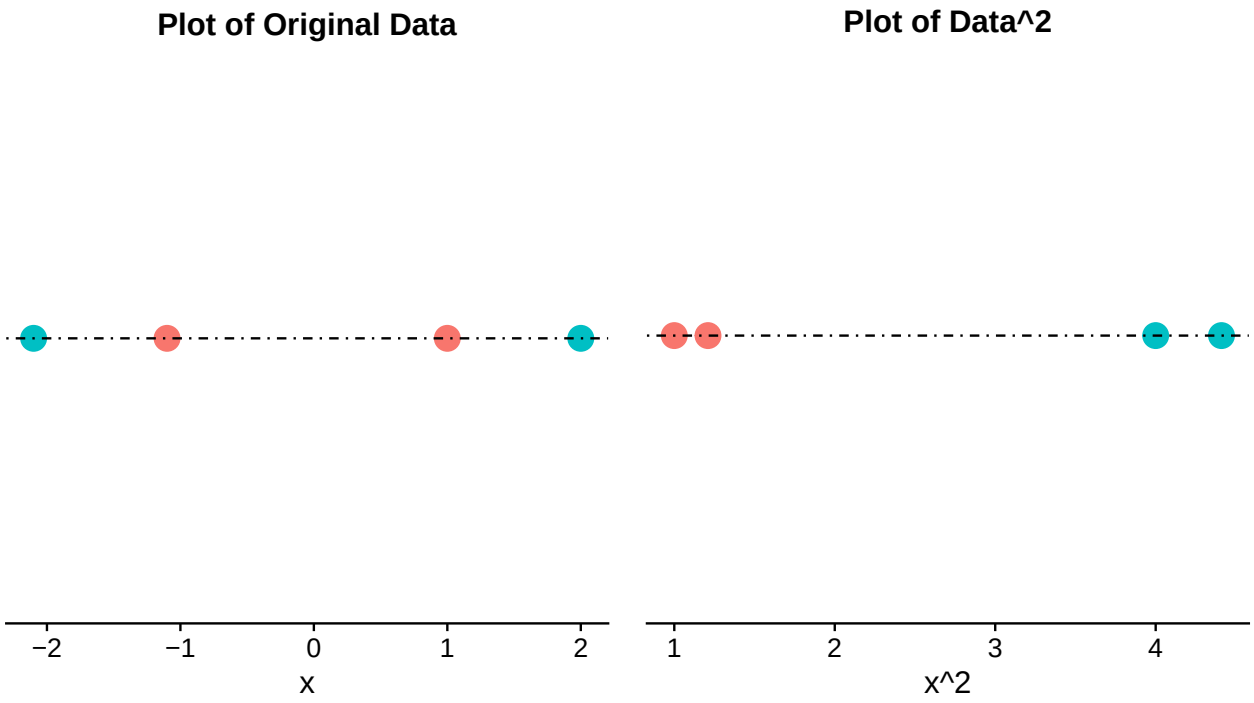
SVM Overview

- SVM is a type of classifier (can also be used for regression)
- Predict binary y from covariates X.
- Different from clustering: in clustering, only have covariates X, no labels y.
- Can run SVM on our data to predict cancer status.
- `kernel = "linear"` means SVM draws a linear decision boundary. Will this work for our data?

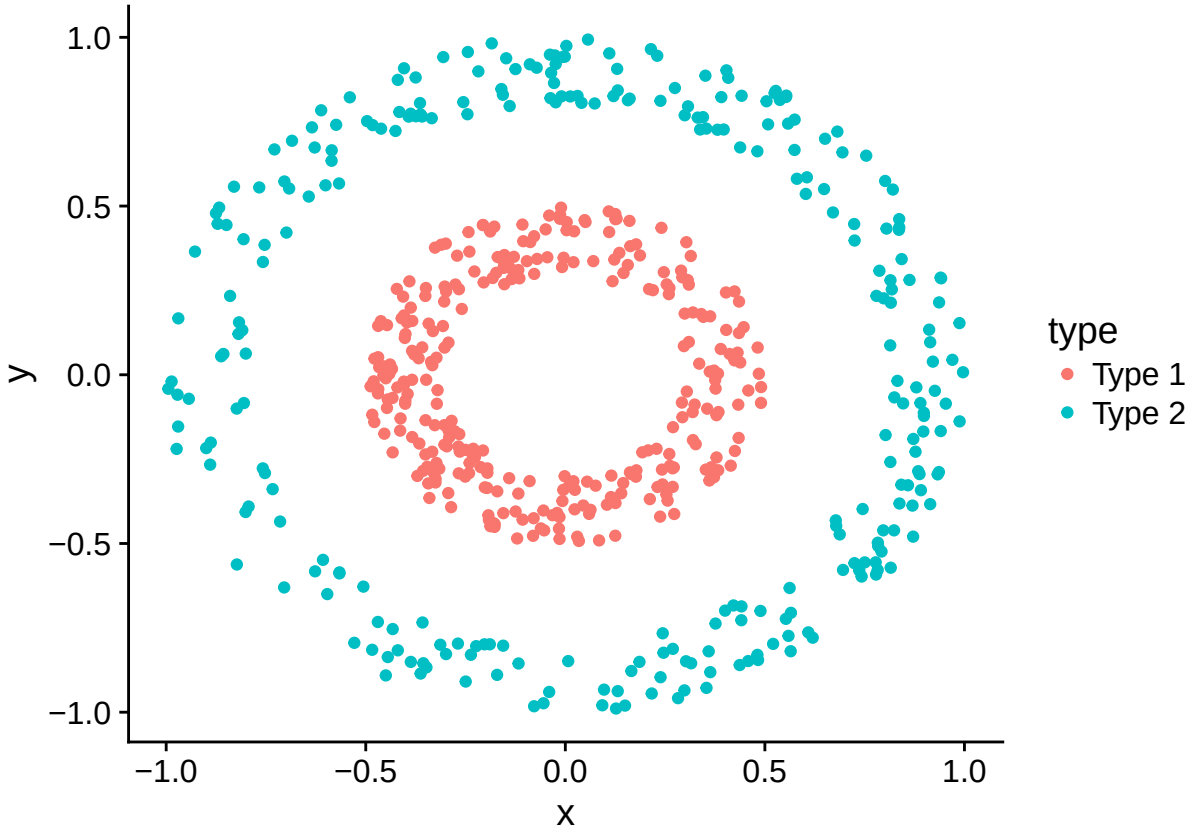
```r
svm_result <- svm(t(combat_edata), as.factor(pheno$hasCancer),
                  kernel = "linear")
svm_result$fitted[1:5]
# just a way to visualize the result, not needed for HW2
confusionMatrix(svm_result$fitted, as.factor(pheno$hasCancer))$table
```

SVM: Nonlinear

- Power of SVM comes in when we use different kernels
- Example: SVM can also draw circular decision boundaries.
- Intuition:

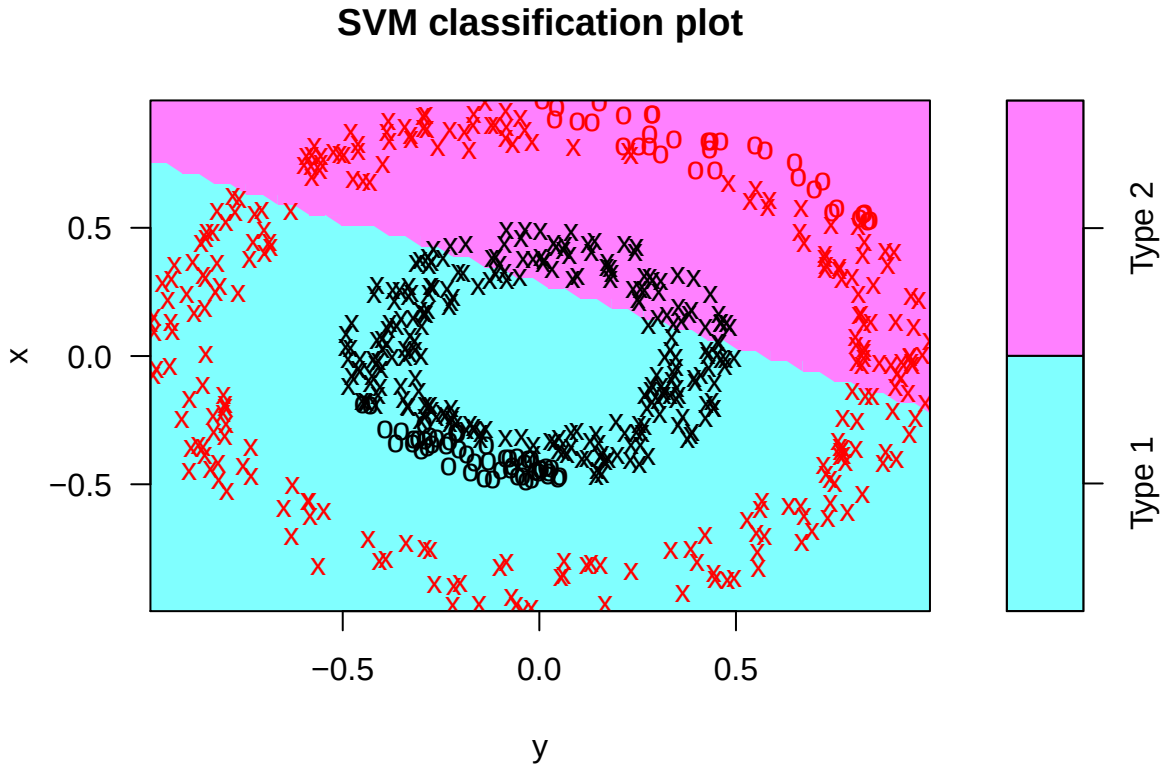


SVM: Nonlinear example



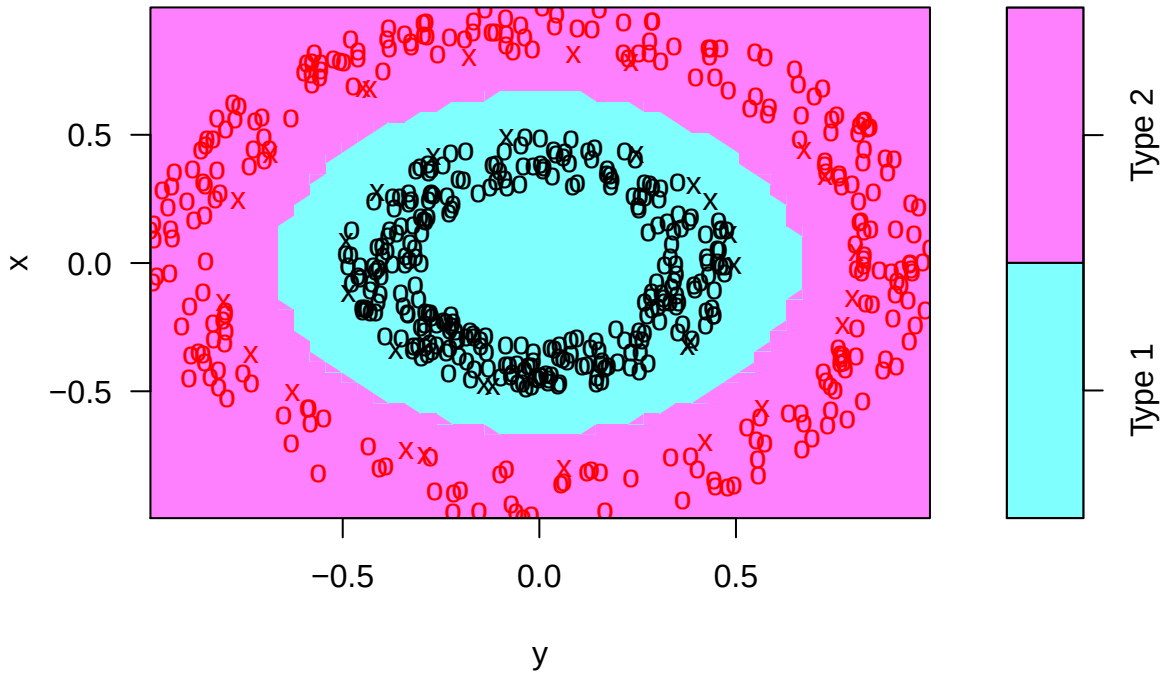
SVM: Using Linear kernel

```
##           Reference
## Prediction Type 1 Type 2
##     Type 1    231    145
##     Type 2     67    136
```



SVM: Using Radial Kernel

```
##            Reference
## Prediction Type 1 Type 2
##     Type 1    298      0
##     Type 2      0    281
```



BWA

- 3 different versions:
  - BWA-backtrack (`aln/samse/sampe`)
  - BWA-SW (`bwasw`)
  - BWA-MEM (`mem`)
- Different recommendations depending on data:
  - BWA-backtrack: Illumina reads up to 100bp
  - BWA-SW / BWA-MEM: longer sequeces from 70bp to 1Mbp
  - BWA-MEM latest version, recommended since generally faster and more accurate
  - BWA-MEM better than BWA-backgrack for 70-100bp Illumina reads
- How to run:

```
bwa aln index.fa sample.fastq > sample.sai
bwa samse index.fa sample.1M.sai sample.fastq > sample.sam
samtools flagstat sample.sam
```

Odyssey

- What is Odyssey? Lots of computers stringed together
- Advantage: More storage, can run many things in parallel (e.g. use 10 computers to process 10 samples at a time)
- Disadvantage: a lot of overhead to get things to work (have to make sure your stuff doesn't interfere with other people's stuff)
- Can't just run stuff through the terminal on Odyssey—login node.
- Have to submit job using srun or sbatch (preferred)
- My tip: start off by requesting very few resources and doing a test run on a small file.

Odyssey Logistics

- Login using ssh (Mac/Linux) or PuTTY (Windows)
- Transfer files using Filezilla
- Details: https://www.rc.fas.harvard.edu/resources/odyssey-quickstart-guide/
- Matt will discuss more next week.

Example Submission Script

- Save this to a file, e.g. `submit.sbatch`.
- Submit by running `sbatch submit.sbatch`.

```
#!/bin/bash
#SBATCH -n 1 # Number of cores requested
#SBATCH -N 1 # Ensure that all cores are on one machine
#SBATCH -t 15 # Runtime in minutes
#SBATCH -p serial_requeue # Partition to submit to
#SBATCH --mem=100 # Memory per cpu in MB (see also --mem-per-cpu)
#SBATCH --open-mode=append
#SBATCH -o output_%j.out # Standard out goes to this file
#SBATCH -e error_%j.err # Standard err goes to this file

LOAD_MODULES
# example:
module load bwa
module load samtools
YOUR_COMMANDS_HERE
```