Introduction

SUPPORT VECTOR MACHINES IN R

Kailash Awati
Instructor

Preliminaries

- **Objective**: gain understanding of how SVMs work; options available in the algorithm and situations in which they work best.
- **Prerequisites:** Intermediate knowledge of R; basic visualization using `ggplot()` .
- **Approach:** Start with 1-dimensional example and gradually move on to more complex examples.

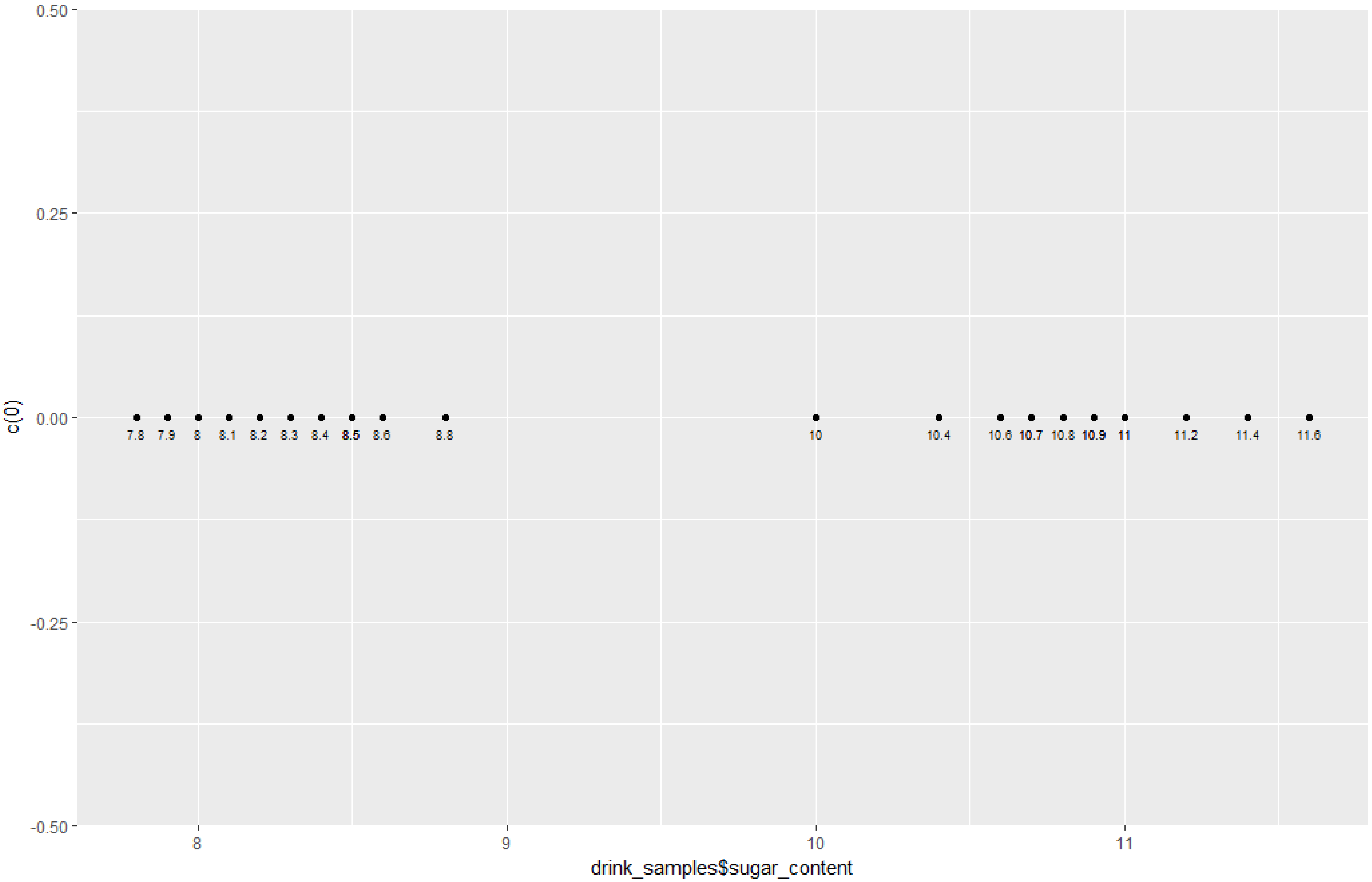
Sugar content of soft drinks

- Soft drink manufacturer has two versions of flagship brand:
  - **Choke** - sugar content 11g/ 100 ml.
  - **Choke-R** - sugar content 8 g/ 100 ml.
- Actual sugar content varies in practice.
- Given 25 samples chosen randomly, find a decision rule to determine brand.
- First step: visualize data!

Sugar content of soft drinks - visualization code

- Data in `drink_samples` dataframe.

```
# Specify dataframe, set plot aesthetics in geom_point (note y = 0)
p <- ggplot(drink_samples) +
  geom_point(aes(sugar_content, 0))
# Label each point with sugar content value, adjust text size and location
p <- p +
  geom_text(aes(sugar_content, 0, label = sugar_content),
            size = 2.5,
            vjust = 2,
            hjust = 0.5)
# Display plot
p
```


0.50
0.25
0.00
-0.25
-0.50
c(0)
7.8
7.9
8
8.1
8.2
8.3
8.4
8.5
8.6
8.8
10
10.4
10.6
10.7
10.8
10.9
11
11.2
11.4
11.6
8
9
10
11
drink_samples$sugar_content

Decision boundaries

- Let's pick two points in the interval as candidate boundaries:
  - 9.1 g/100 ml
  - 9.7 g/100 ml
- Classification (decision) rules:
  - if (y < 9.1) then "Choke-R" else "Choke"
  - if (y < 9.7) then "Choke-R" else "Choke"
- Let's visualize them on the plot shown on the previous slide.

Decision boundaries - visualization code

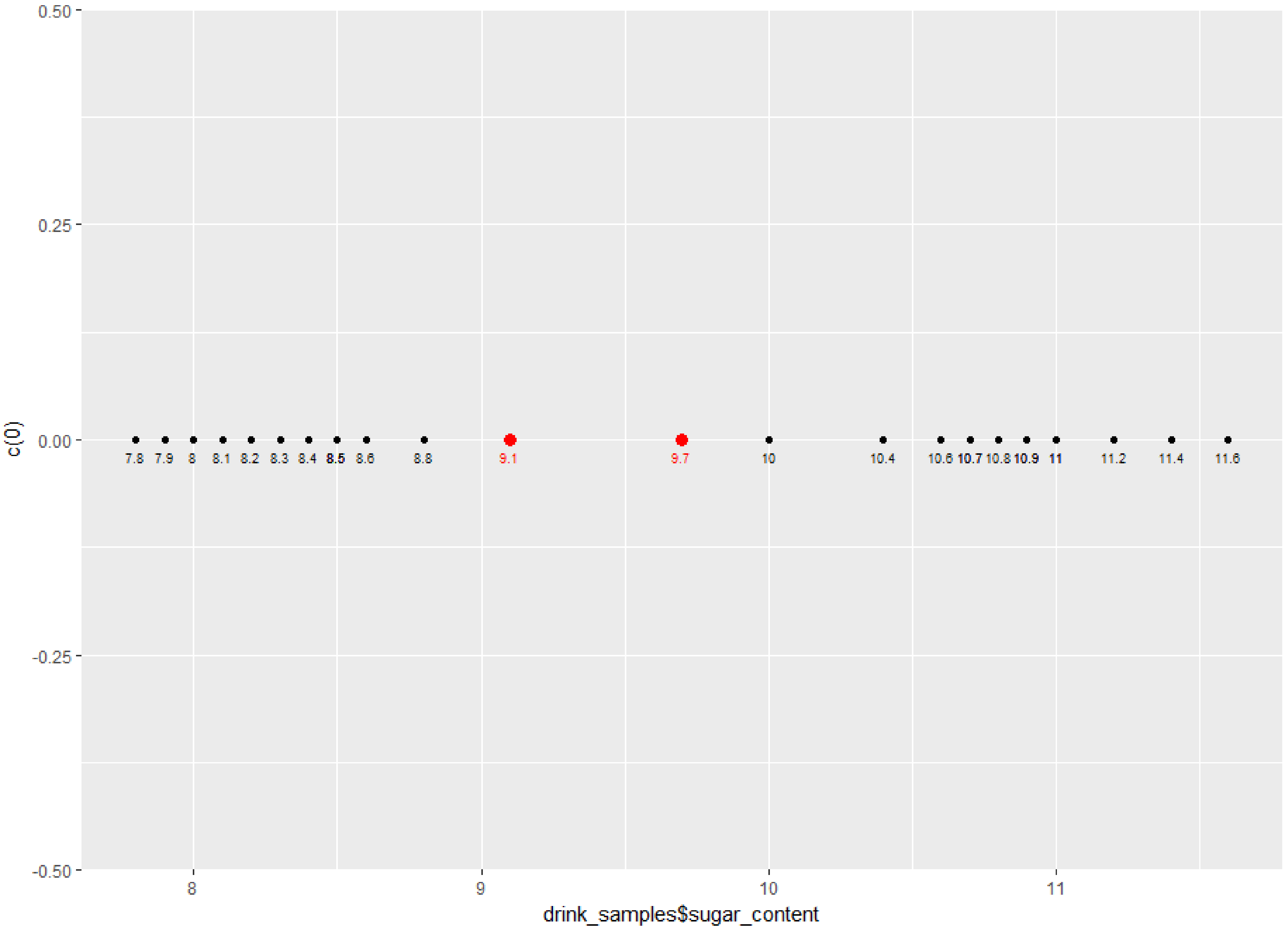
- Create a dataframe containing the two decision boundaries.

```
# Define data frame containing decision boundaries
d_bounds <- data.frame(sep = c(9.1, 9.7))
```

Decision boundaries - visualization code

- Add to plot using `geom_point()`

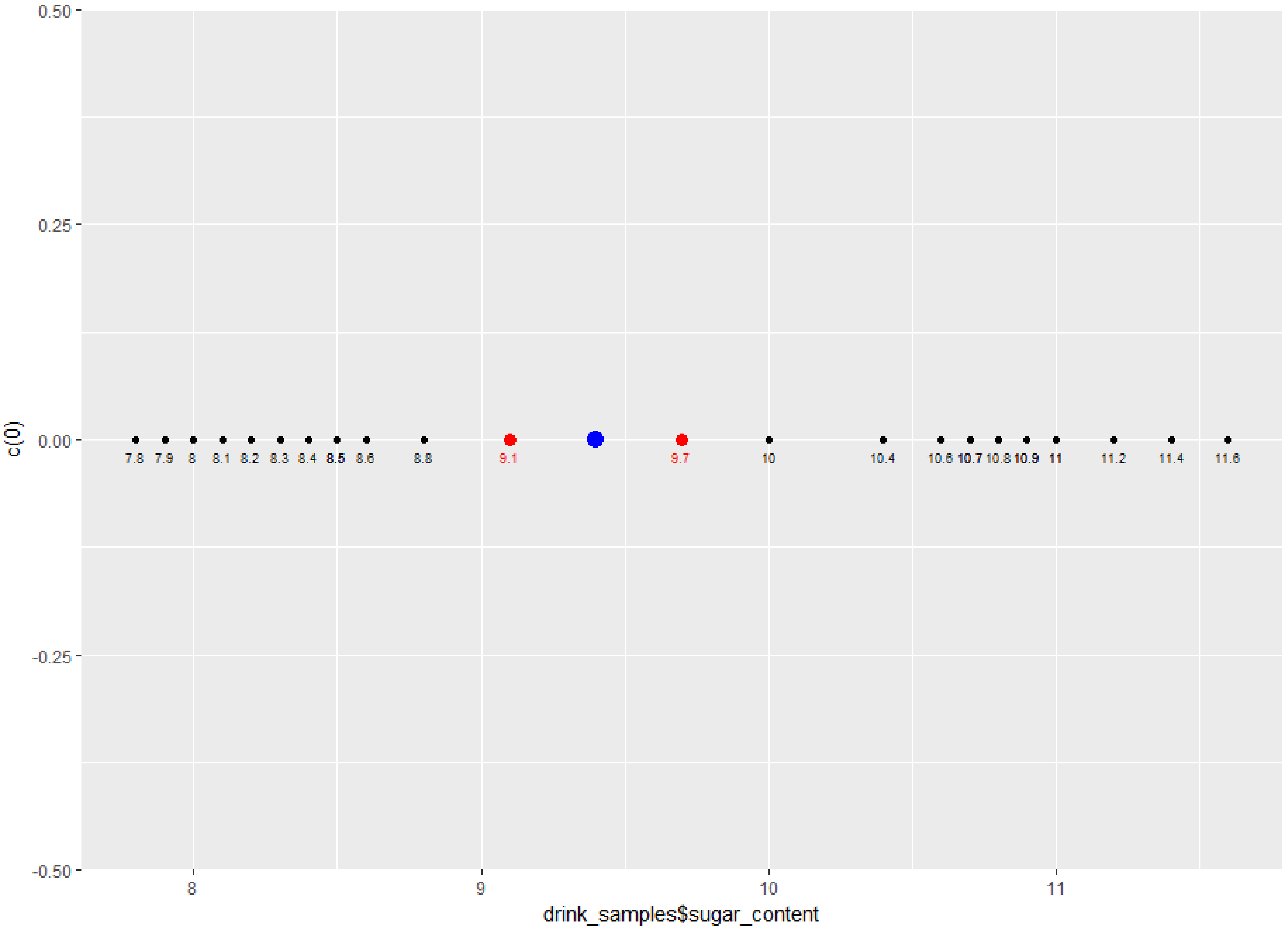
```
# Add decision boundaries to previous plot
p <- p +
  geom_point(data = d_bounds,
             aes(sep, 0),
             color = "red",
             size = 3) +
  geom_text(data = d_bounds,
            aes(sep, 0, label = sep),
            size = 2.5,
            vjust = 2,
            hjust = 0.5,
            color = "red")
# Display plot
p
```


0.50
0.25
0.00
-0.25
-0.50
c(0)
7.8 7.9 8 8.1 8.2 8.3 8.4 8.5 8.6 8.8 9.1 9.7 10 10.4 10.6 10.7 10.8 10.9 11 11.2 11.4 11.6
8
9
10
11
drink_samples$sugar_content

Maximum margin separator

- The best decision boundary is one that maximizes the margin: **maximal margin separator**
- Maximal margin separator lies halfway between the two clusters.
- Visualize the maximal margin separator.

```
# Create data frame with maximal margin separator
mm_sep <- data.frame(sep = c((8.8 + 10) / 2))
# Add mm boundary to previous plot
p <- p +
  geom_point(data = mm_sep,
             aes(sep, 0),
             color = "blue",
             size = 4)
# Display plot
p
```



0.50
0.25
0.00
-0.25
-0.50
c(0)
7.8
7.9
8
8.1
8.2
8.3
8.4
8.5
8.6
8.8
9.1
9.7
10
10.4
10.6
10.7
10.8
10.9
11
11.2
11.4
11.6
8
9
10
11
drink_samples$sugar_content

Time to practice!

SUPPORT VECTOR MACHINES IN R

Generating a linearly separable dataset

SUPPORT VECTOR MACHINES IN R

Kailash Awati
Instructor

Overview of lesson

- Create a dataset that we'll use to illustrate key principles of SVMs.
- Dataset has two variables and a linear decision boundary.

Generating a two-dimensional dataset using runif()

- Generate a two variable dataset with 200 points
- Variables `x1` and `x2` uniformly distributed in (0,1).

```
# Preliminaries...
# Set required number of data points
n <- 200
# Set seed to ensure reproducibility
set.seed(42)
# Generate dataframe with two predictors x1 and x2 in (0,1)
df <- data.frame(x1 = runif(n),
                 x2 = runif(n))
```

Creating two classes

- Create two classes, separated by the straight line decision boundary `x1 = x2`
- Line passes through (0, 0) and makes a 45 degree angle with horizontal
- Class variable `y = -1` for points below line and `y = 1` for points above it

```
# Classify points as -1 or +1
df$y <- factor(ifelse(df$x1 - df$x2 > 0, -1, 1),
               levels = c(-1, 1))
```

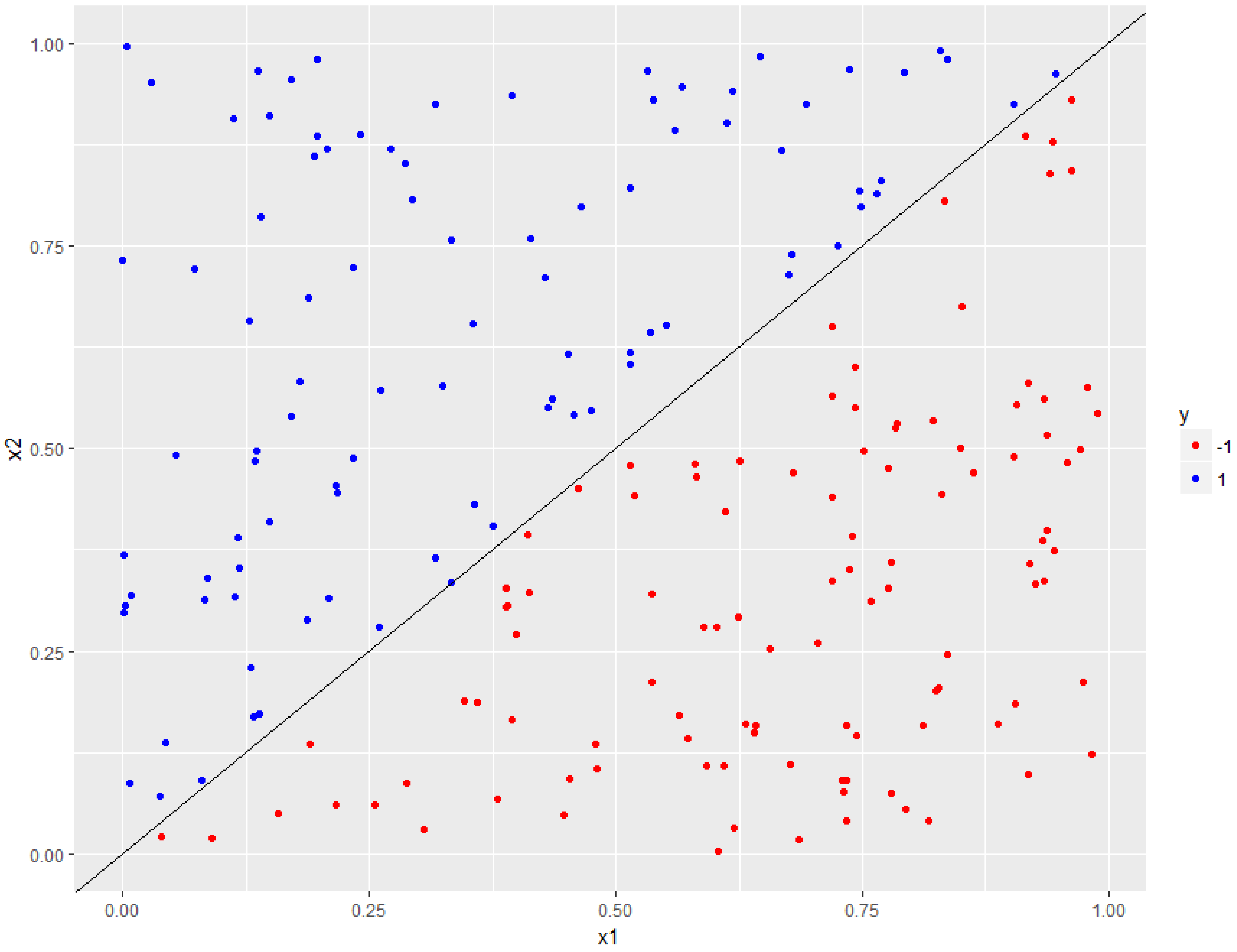
Visualizing dataset using ggplot

- Create 2 dimensional scatter plot with x1 on the x axis and x2 on the y-axis
- Distinguish classes by color (below line = red; above line = blue)
- Decision boundary is line `x1 = x2` : passes through (0, 0) and has slope = 1

```
library(ggplot2)

# Build plot
p <- ggplot(data = df, aes(x = x1, y = x2, color = y)) +
     geom_point() +
     scale_color_manual(values = c("-1" = "red", "1" = "blue")) +
     geom_abline(slope = 1, intercept = 0)

# Display it
p
```

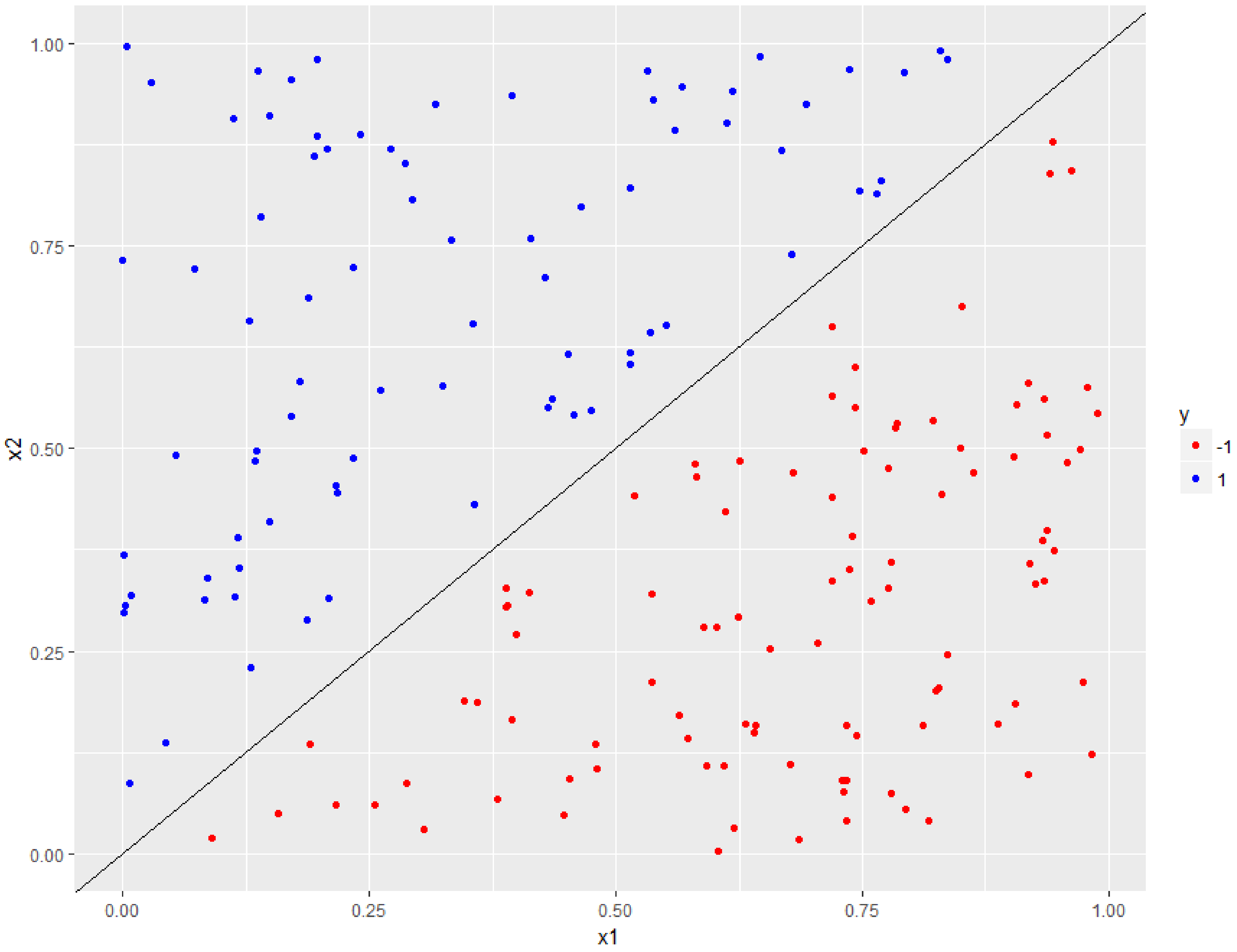


1.00
0.75
0.50
0.25
0.00
x2
0.00
0.25
0.50
0.75
1.00
x1
y
-1
1

Introducing a margin

- To create a margin we need to remove points that lie close to the boundary
- Remove points that have x1 and x2 values that differ by less than a specified value

```
# Create a margin of 0.05 in dataset
delta <- 0.05
# Retain only those points that lie outside the margin
df1 <- df[abs(df$x1 - df$x2) > delta, ]
# Check number of data points remaining
nrow(df1)
# Replot dataset with margin (code is exactly same as before)
p <- ggplot(data = df1, aes(x = x1, y = x2, color = y)) +
     geom_point() +
     scale_color_manual(values = c("red", "blue")) +
     geom_abline(slope = 1, intercept = 0)
# Display plot
p
```

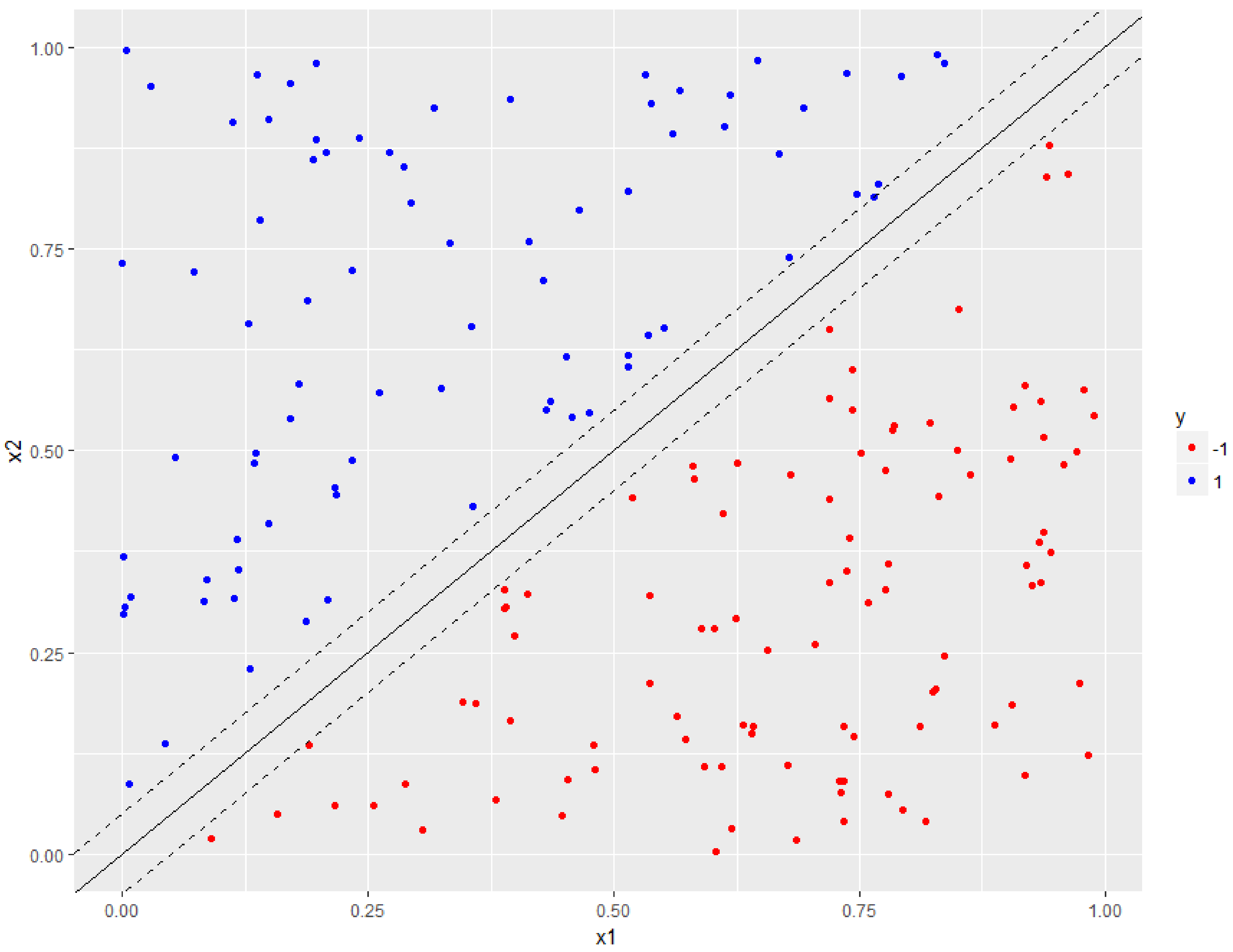

1.00
0.75
0.50
0.25
0.00
x2
0.00
0.25
0.50
0.75
1.00
x1
y
-1
1

Plotting the margin boundaries

- The margin boundaries are:
  - parallel to the decision boundary (slope = 1).
  - located delta units on either side of it (delta = 0.05).

```
p <- p +
     geom_abline(slope = 1, intercept = delta, linetype = "dashed") +
     geom_abline(slope = 1, intercept = -delta, linetype = "dashed")


p
```



1.00
0.75
0.50
0.25
0.00
x2
0.00
0.25
0.50
0.75
1.00
x1
y
-1
1

Time to practice!

SUPPORT VECTOR MACHINES IN R