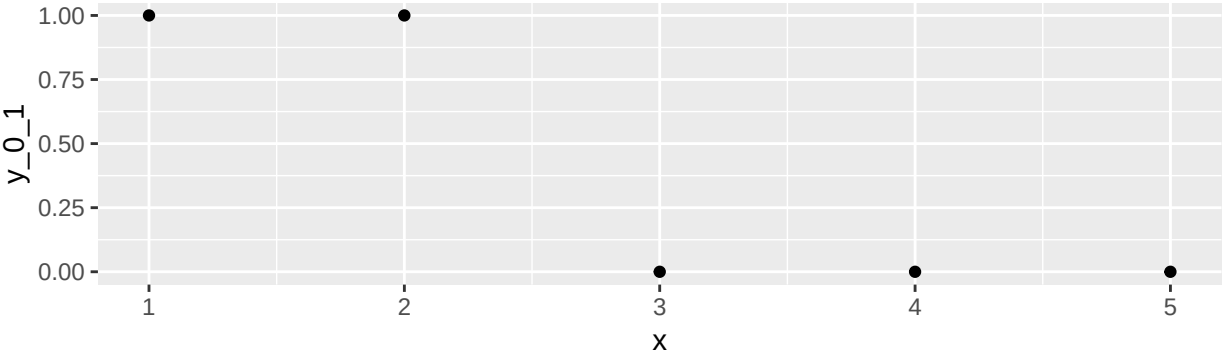
KNN Classification

Suppose we have the following data:

```
example_data
```

```
##   x y y_0_1
## 1 1 B     1
## 2 2 B     1
## 3 3 A     0
## 4 4 A     0
## 5 5 A     0
```

```
ggplot(data = example_data, mapping = aes(x = x, y = y_0_1)) +
  geom_point()
```



2 Basic Approaches to Estimating $p_1(x)$:

1. Specify a model like $p_1(x) = \frac{e^{\beta_0+\beta_1 x}}{1+e^{\beta_0+\beta_1 x}}$. Estimate the *parameters* β_0 and β_1.
2. Local Approach: $p_1(x_0)$ should look like the data in a neighborhood of x_0

Models that don't specify a specific parametric form for p are often called *nonparametric*.

K Nearest Neighbors

For class j, $\hat{p_j}(x_0) = \frac{1}{K}\sum_{i \in N_0^{(k)}} I(y_i = j)$

Here $N_0^{(k)}$ is a set of indices for the k observations that have values x_i nearest to the test point x_0.

Example:

- Suppose $x_0 = 2.25$
- Set $k = 3$
- In our training data set, the k nearest neighbors are $i = 1$, $i = 2$, and $i = 3$. $N_0^{(3)} = \{1, 2, 3\}$
- Our estimated probability that $Y_0 = 1$ given that $x_0 = 2.25$ is the proportion of responses for those three observations that are equal to 1:

```
# we will never actually compute f_hat like this in practice!
neighborhood_indices <- c(1, 2, 3)
p_1_hat <- mean(example_data$y_0_1[neighborhood_indices] == 1)
p_1_hat
```

```
## [1] 0.6666667
```

In this case, the estimated class is $\hat{Y}_0 = 1$.

Prediction with `FNN::knn`

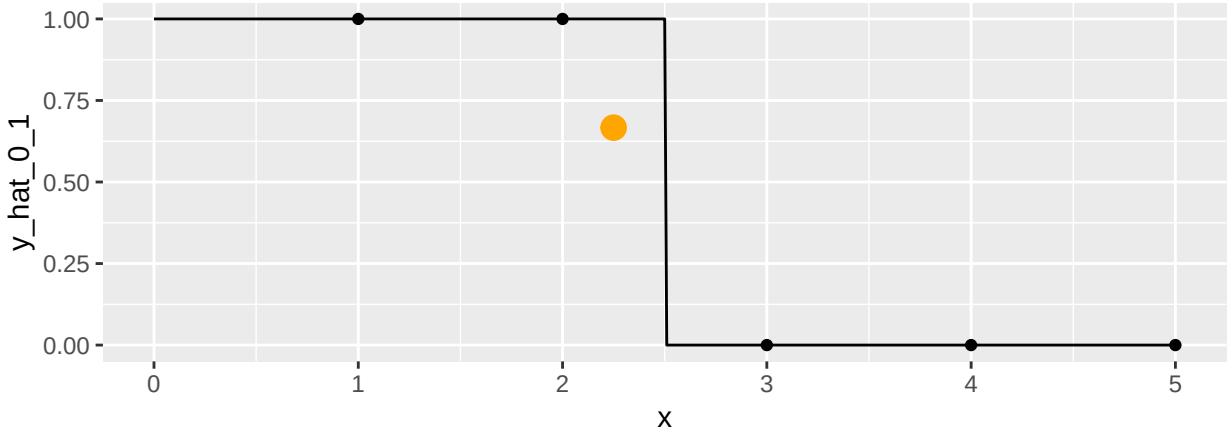
The `knn` function in the `FNN` package will find the predicted class and its estimated probability for us if we provide training data and a vector of test values for x.

Annoyingly, it won't return the full estimated function $\widehat{p}_1$; only the estimated probability for the most likely class.

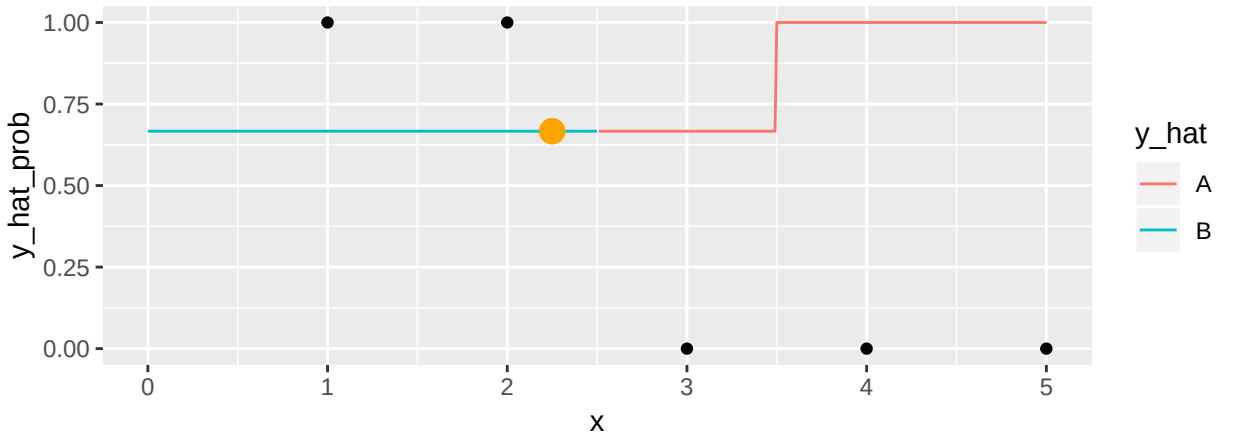
```
library(FNN)
f_hat_data <- data.frame(
  x = seq(from = 0, to = 5, by = 0.01)
)
head(f_hat_data)
```

```
##      x
## 1 0.00
## 2 0.01
## 3 0.02
## 4 0.03
## 5 0.04
## 6 0.05
```

```
knn_predictions <- knn(
  train = example_data %>% dplyr::select(x), # a data frame with train set explanatory variables
  test = f_hat_data, # a data frame with test set explanatory variables
  cl = example_data$y, # a vector (not data frame!) with train set response variable; OK to use factor
  k = 3, # value of k for k nearest neighbors
  prob = TRUE) # request to return estimated class membership probability

f_hat_data <- f_hat_data %>%
  mutate(
    y_hat = knn_predictions,
    y_hat_0_1 = as.numeric(y_hat) - 1,
    y_hat_prob = attr(knn_predictions, "prob")
  )

ggplot() +
  geom_line(data = f_hat_data, mapping = aes(x = x, y = y_hat_0_1)) +
  geom_point(data = example_data, mapping = aes(x = x, y = y_0_1)) +
  geom_point(mapping = aes(x = 2.25, y = 2/3), color = "orange", size = 4)
```




```
ggplot() +
  geom_line(data = f_hat_data, mapping = aes(x = x, y = y_hat_prob, color = y_hat)) +
  geom_point(data = example_data, mapping = aes(x = x, y = y_0_1)) +
  geom_point(mapping = aes(x = 2.25, y = 2/3), color = "orange", size = 4)
```



Nonlinear Decision Boundaries; Flexibility is Determined by k

Returning to the `Default` data set about credit card debt, let's try to predict whether a customer will default on their credit card debt using their account `balance` and `income`.

Recall that logistic regression gave us a linear decision boundary. Let's compare with KNN, where we have rescaled the balance and income.

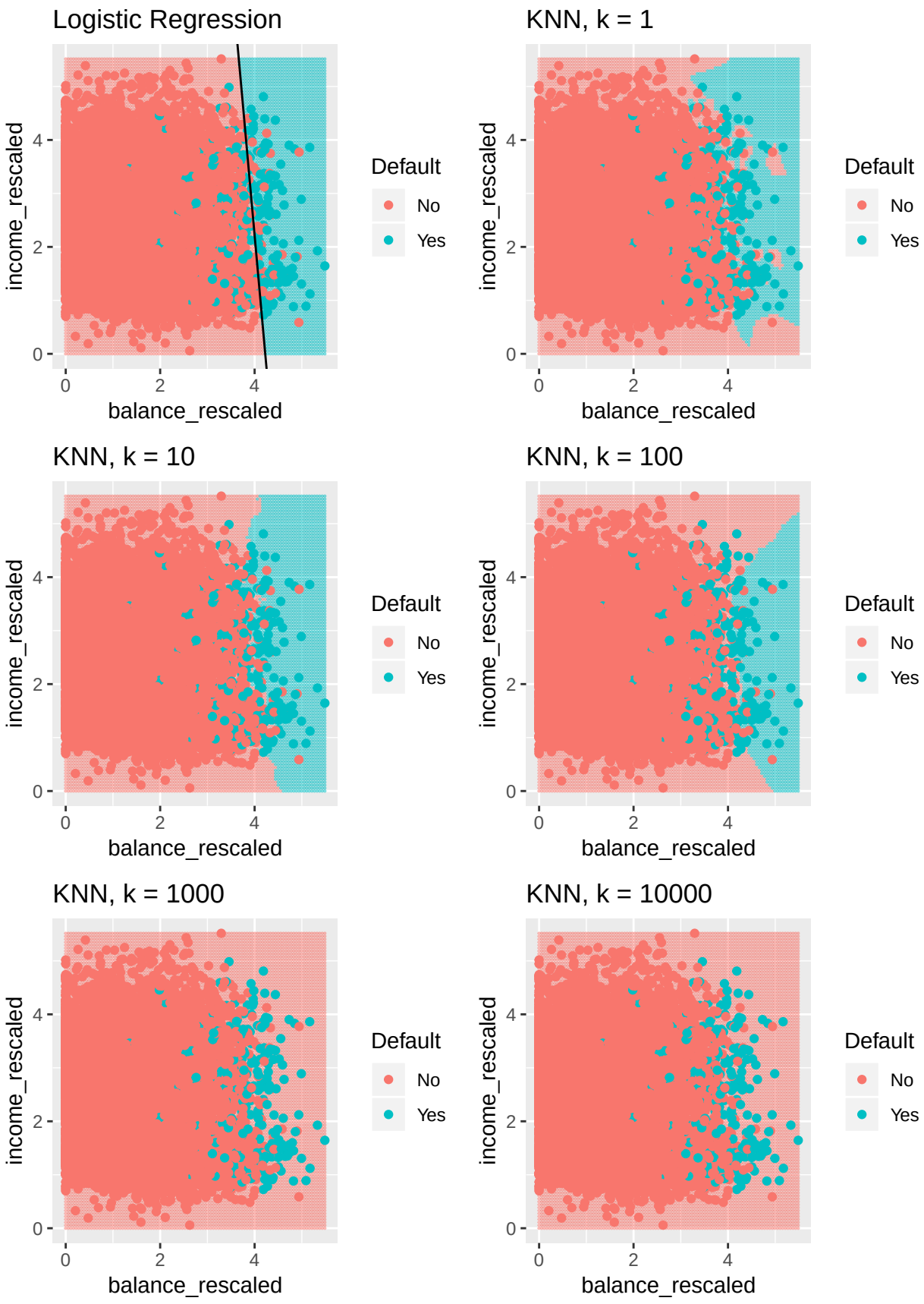
```
library(ISLR)

Default <- Default %>%
  mutate(
    balance_rescaled = balance / sd(balance),
    income_rescaled = income / sd(income)
  )
```

```
fit <- glm(default ~ balance_rescaled + income_rescaled, data = Default, family = binomial)
summary(fit)
```

```
##
## Call:
## glm(formula = default ~ balance_rescaled + income_rescaled, family = binomial,
##     data = Default)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.4725  -0.1444  -0.0574  -0.0211   3.7245
##
## Coefficients:
##                   Estimate Std. Error z value Pr(>|z|)
## (Intercept)      -11.54047    0.43476 -26.545  < 2e-16 ***
## balance_rescaled   2.73159    0.10998  24.836  < 2e-16 ***
## income_rescaled    0.27752    0.06649   4.174 2.99e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2920.6  on 9999  degrees of freedom
## Residual deviance: 1579.0  on 9997  degrees of freedom
## AIC: 1585
##
## Number of Fisher Scoring iterations: 8
```

```
max_balance_rescaled <- max(Default$balance_rescaled)
max_income_rescaled <- max(Default$income_rescaled)
```

```r
background <- expand.grid(
    balance_rescaled = seq(from = 0, to = max_balance_rescaled, length = 101),
    income_rescaled = seq(from = 0, to = max_income_rescaled, length = 101))


background_logistic <- background %>%
  mutate(
    est_prob_default = predict(fit, newdata = background, type = "response"),
    est_default = ifelse(est_prob_default > 0.5, "Yes", "No")
  )

p_logistic <- ggplot() +
  geom_point(data = background_logistic,
    mapping = aes(x = balance_rescaled, y = income_rescaled, color = est_default), size = 0.1, alpha = 0.5) +
  geom_point(data = Default, mapping = aes(x = balance_rescaled, y = income_rescaled, color = default)) +
  scale_color_discrete("Default") +
  geom_abline(intercept = 11.54047 / 0.27752, slope = - 2.73159 / 0.27752) +
  ggtitle("Logistic Regression")

get_knn_plot <- function(k) {
  knn_predictions <- knn(
    train = Default %>% dplyr::select(balance_rescaled, income_rescaled), # a data frame with train set expla
    test = background, # a data frame with test set explanatory variables
    cl = Default$default, # a vector (not data frame!) with train set response variable; OK to use factor
    k = k, # value of k for k nearest neighbors
    prob = TRUE) # request to return estimated class membership probability

  background_knn <- background %>%
    mutate(
      est_default = knn_predictions
    )

  p_k <- ggplot() +
    geom_point(data = background_knn,
      mapping = aes(x = balance_rescaled, y = income_rescaled, color = est_default), size = 0.1, alpha = 0.5)
    geom_point(data = Default, mapping = aes(x = balance_rescaled, y = income_rescaled, color = default)) +
    scale_color_discrete("Default") +
    geom_abline(intercept = 1.154e+01 / 2.081e-05, slope = - 5.647e-03 / 2.081e-05) +
    ggtitle(paste0("KNN, k = ", k))

  return(p_k)
}

plots_knn <- lapply(c(1, 10, 100, 1000, 10000), get_knn_plot)

grid.arrange(p_logistic, plots_knn[[1]], plots_knn[[2]], plots_knn[[3]], plots_knn[[4]], plots_knn[[5]], ncol
```



See Chapter 5 (coming soon!) for formal approaches to choosing k.