Class 9: Linear Regression 3 and Intro to Classification

MGSC 310

Prof. Jonathan Hersh

Class 9: Announcements

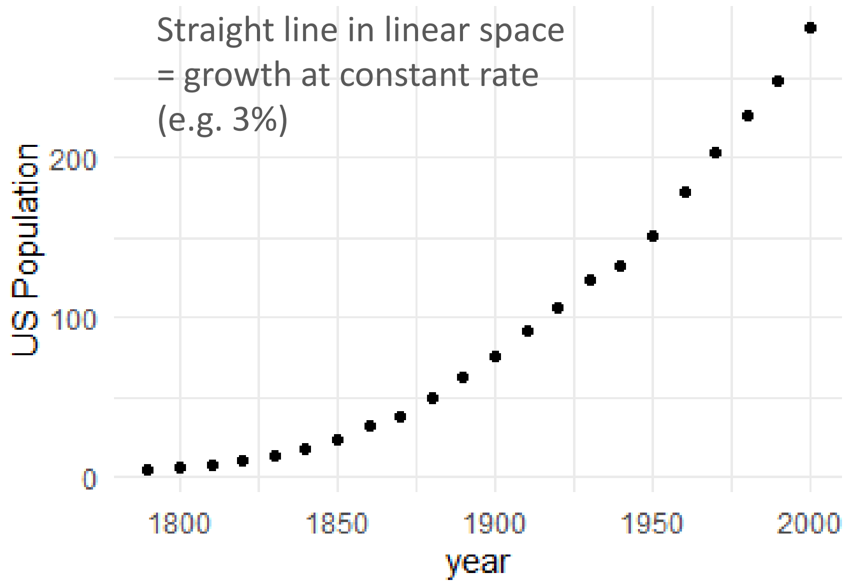
1. TA Office Hours:
   - Tuesdays: 5:30 – 7
   - Thursdays: 12:30-2
   - Mondays: 5-6:30
2. Quiz 3 will post tonight, due Thursday @ midnight
3. Problem Set 2 Posted, Due Sept 29
   - Late problem sets docked 10% per day unless extenuating circumstances

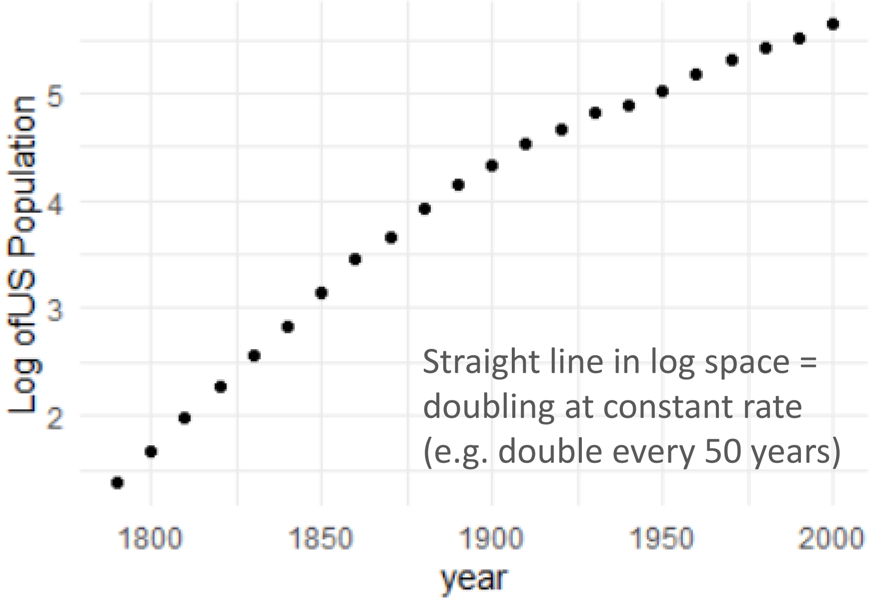
1. Grades for Psets
   - Aim to grade in 1 week
   - Not always feasible given my and TA schedule
   - Pset 1 grades out later this week

Class 9: Outline

1. Log Transformations and Interpreting log-log Regressions
2. Model Evaluation
    - Testing and Training Sets, RMSE, and R-Squared
3. Lab Class 9
4. Why Classification Models?
5. Logistic Function
6. Log Odds Ratio

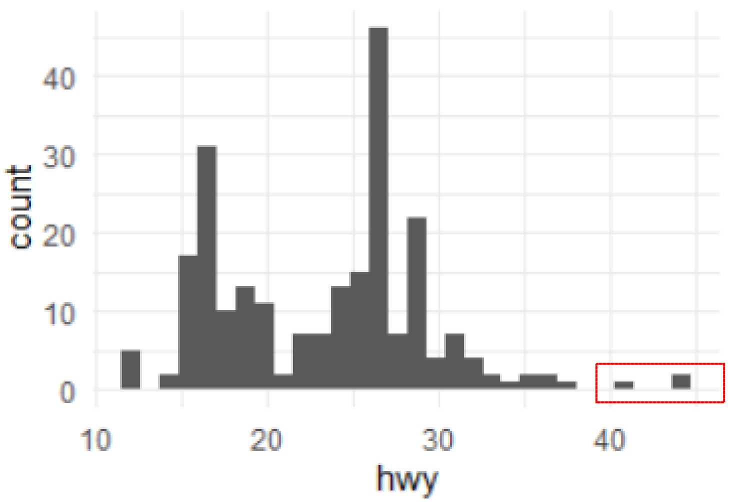
Log Transformations

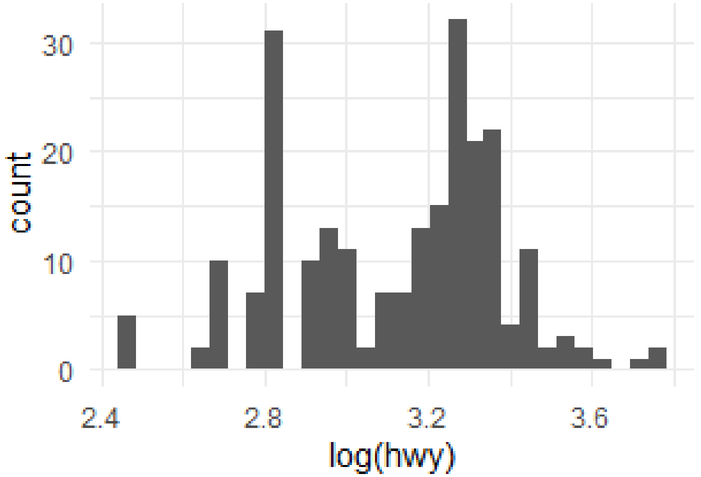


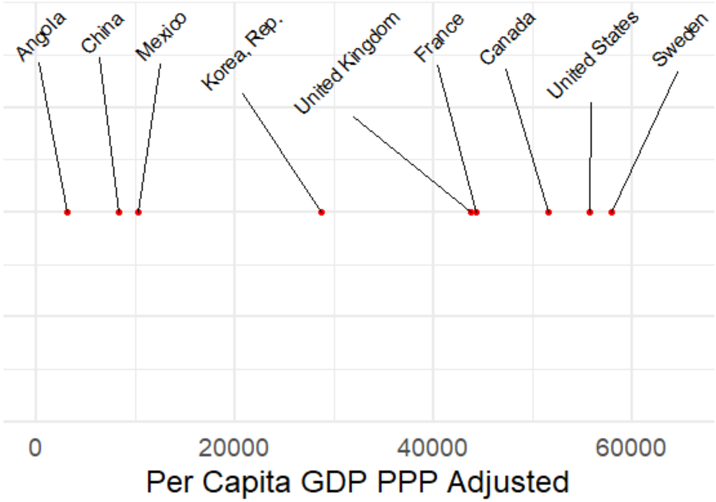


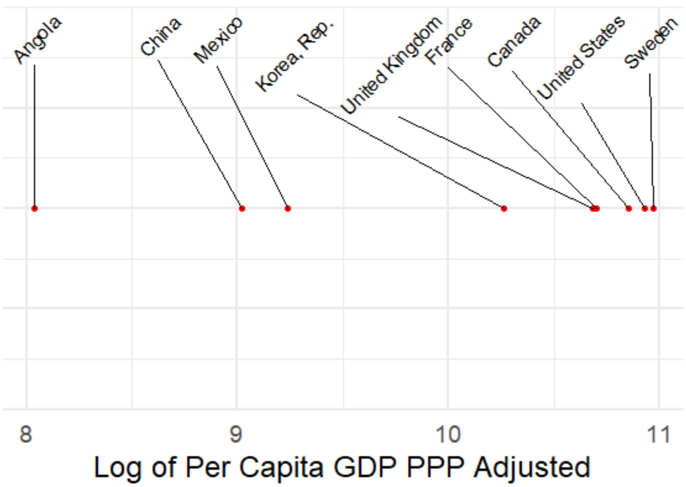
- Recall that log is the inverse of exponentiation
- Logging a number answers the question "what is the exponent I must raise the base of the log to produce this number"
    - $\log_{10} 100 = 2 \rightarrow 10^2 = 100$
    - $\log_{10} 1000 = 3 \rightarrow 10^3 = 1000$
- The natural log is $\log_e x = \ln(x)$ where e=2.718…
- Each unit increase of ln(x) = a doubling of x
- This is a useful data transformation because we often want to incorporate data that increases rapidly in our regression analysis

Why Log Transformations For Regression Data









- Log transforming our data is often very useful if our data is particularly “spread out”
- In particular if there are outlier values this will improve model performance
- Some data like income should usually be logged

Model accuracy often improves when logging dependent variable but interpretation can suffer

Log-Log Regression Model Coefficients = Elasticities!

$$\log(hwy_i) = \beta_0 + \beta_1 \log(displ_i) + \beta_2 year_i + \epsilon_i$$

```
> mod1 <- lm(log(hwy) ~ log(displ) + year,
+            data = mpg)
> summary(mod1)

Call:
lm(formula = log(hwy) ~ log(displ) + year, data = mpg)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.46033 -0.09856 -0.00202  0.08837  0.48681 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -10.874596   4.551216  -2.389  0.01768 *  
log(displ)   -0.560514   0.027052 -20.720  < 2e-16 ***
year          0.007314   0.002274   3.217  0.00148 ** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1547 on 231 degrees of freedom
Multiple R-squared:  0.6502,	Adjusted R-squared:  0.6471 
F-statistic: 214.7 on 2 and 231 DF,  p-value: < 2.2e-16
```

- Suppose we log our hwy regression model
- How do we now interpret the coefficient $\beta_{\log(displ)}$
  - $y = \exp(\beta_0 + \beta_1 \log(x) + \epsilon)$
  - $\frac{dy}{dx} = \frac{\beta}{x} \exp(\beta_0 + \beta_1 \log(x) + \epsilon)$
  - $\Rightarrow \beta_1 = \frac{dy/y}{dx/x}$

Therefore log coefficients can be interpreted as elasticities!

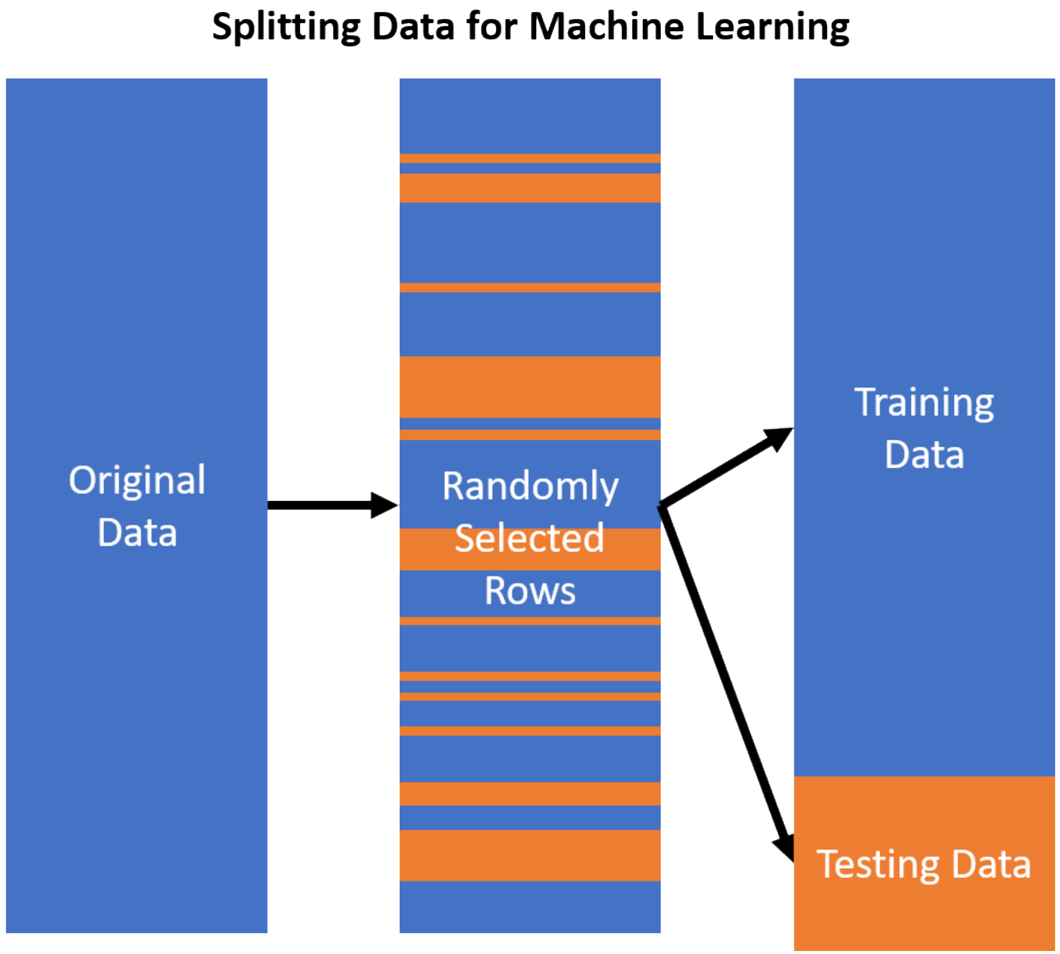
A 1% change in x-variable results in a β_1 % change in the outcome variable

- Here a 1% increase in displacement -> a 0.56% decrease in highway mpg.

Class 9: Outline

1. Log Transformations and Interpreting log-log Regressions
2. **Model Evaluation**
    - Testing and Training Sets, RMSE, and R-Squared
3. Lab Class 9
4. Why Classification Models?
5. Logistic Function
6. Log Odds Ratio

Recall: Training and Testing Sets



Training set: (observation-wise) subset of data used to develop models

Test set: subset of data used during intermediate stages to “tune” model parameters

Building Training and Testing Sets in R

```r
#-------------------------------------------------------------
# Testing and Training Sets
#-------------------------------------------------------------

# install rsample if necessary
# install.packages('rsample')
library('rsample')
library('tidyverse')
data(mpg)

# always want to set a seed before doing any randomization
# procedure to make our code reproducible
set.seed(1818)

# set train proportion = % of total data in training set
# common values are 0.9, 0.75, 0.8, 0.5
train_prop <- 0.8
mpg_split <- initial_split(mpg, prop = train_prop)
```

```r
> mpg_train <- training(mpg_split)
> mpg_test <- testing(mpg_split)
> # check the number of rows to ensure training
> # and testing split is correct
> nrow(mpg_train)
[1] 188
> nrow(mpg_test)
[1] 46
>
```

- initial_split() is a helper function to create testing and training sets
- Must specify the data frame to split
- Can also specify the % of data to use for training (defaults to 75%)
- The functions training() and testing() will create separate testing and training sets from the original data set

Always set seed before any randomize procedure to ensure code is reproducible

Generating In-Sample (Training) and Out-of-Sample (Test) Predictions

```r
#-----------------------------------------------------------
# Estimating Models on Training Sets
#-----------------------------------------------------------
# estimate a model using the training set
mod <- lm(hwy ~ year + displ,
          data = mpg_train)

# generate in-sample (training) predictions
preds_train <- predict(mod)
# either method works
preds_train <- predict(mod, newdata = mpg_train)

# generate out-of-sample (test set) predictions
preds_test <- predict(mod, newdata = mpg_test)
```

- Estimate a model on the training set
- Never estimate a model using the test set
- **In-sample predictions** are the predicts using data in the training set
- **Out-of-sample predictions** are the predicts using data in the test set

Quantitative Model Evaluation Using Yardstick

- The package 'yardstick' has several functions to quantitatively evaluate model performance
- We must first compile our model output in a 'results' data frame
- We include the predictions from the model
- True outcome values must exclude any missing values for Xs or Ys
- We must also do this for the test data set

```
library('yardstick')

# create a
results_train <- data.frame(
  predicted = preds_train,
  actual = mpg_train %>%
    filter(complete.cases(hwy, year, displ)) %>%
    select(hwy),
  type = rep("train", length(preds_train))
) %>%
  rename(`predicted` = 1, `actual` = 2, `type` = 3)
```

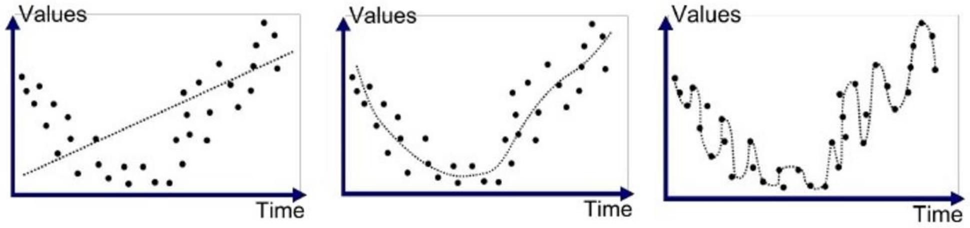
```
results_test <- data.frame(
  predicted = preds_test,
  actual = mpg_test %>%
    filter(complete.cases(hwy, year, displ)) %>%
    select(hwy),
  type = rep("test", length(preds_test))
) %>%
  rename(`predicted` = 1, `actual` = 2, `type` = 3)
```

Overfit Vs Underfit: Compare the Test and Training Error

- rmse() function computes root mean squared error (i.e. MSE^(1/2))
- Pass the data frame of results
- Also the column names (in the results DF) for the predicted and true values
- We compare across models by examining the same error metric
- Since the error in the test set is lower than error in the training set we conclude the model is underfit, meaning we can increase model complexity

```
> rmse(results_train, predicted, actual)
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        4.03
```

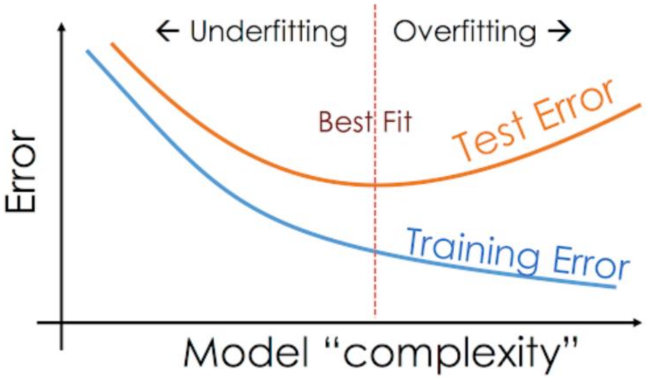
```
> rmse(results_test, predicted, actual)
# A tibble: 1 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard        2.34
```



Underfitted   Good Fit/Robust   Overfitted

Other Functions for Evaluation: metrics() and mae()

```
> metrics(results_train, predicted, actual)
# A tibble: 3 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       4.03
2 rsq     standard       0.563
3 mae     standard       2.93
> metrics(results_test, predicted, actual)
# A tibble: 3 x 3
  .metric .estimator .estimate
  <chr>   <chr>          <dbl>
1 rmse    standard       2.34
2 rsq     standard       0.820
3 mae     standard       1.83
```



- metrics() function estimates a series of evaluation metrics

- mae is "mean absolute error"

$$MAE = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

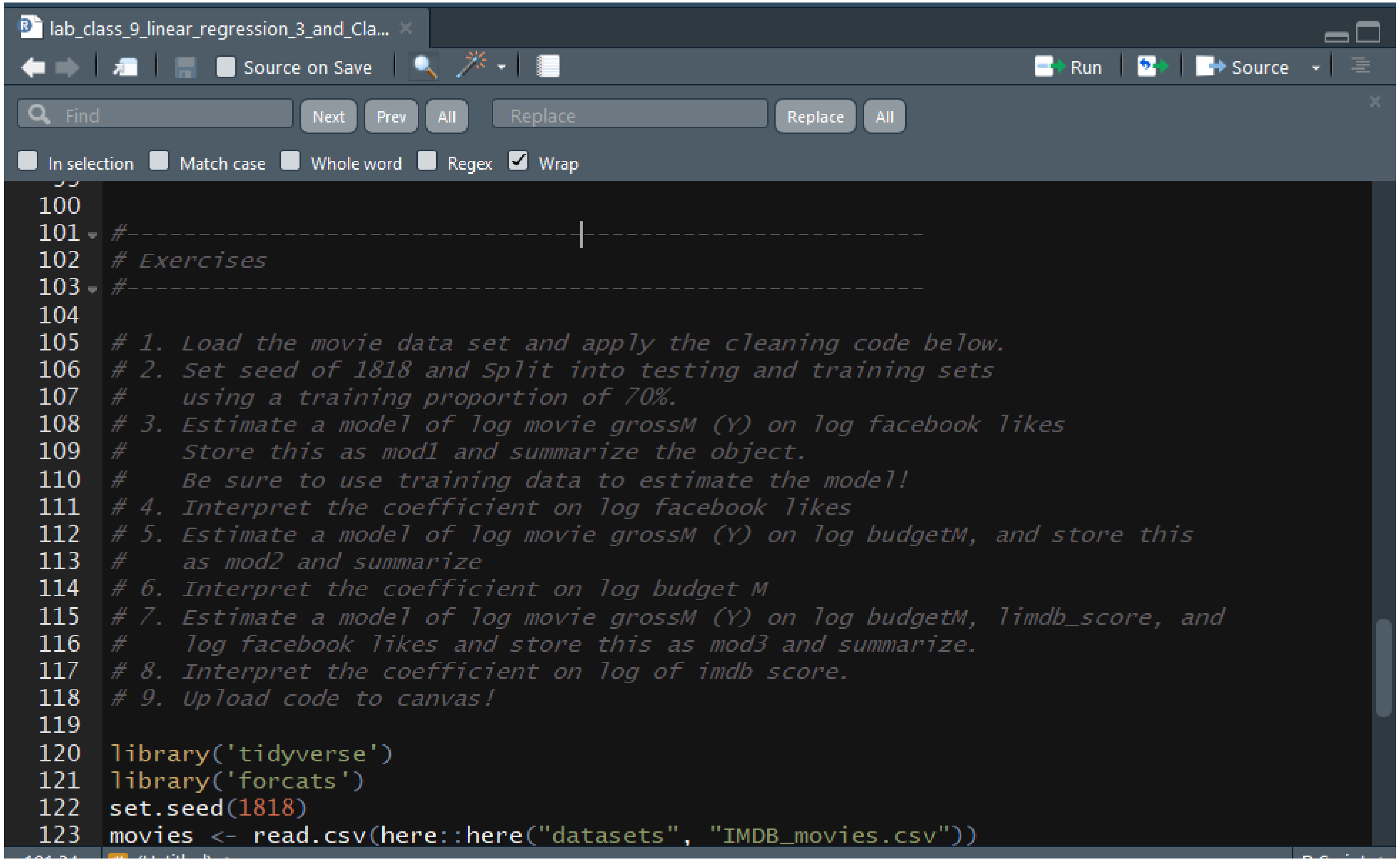
- rsq is our friend R^2

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y}_i)^2}{\sum_i (y_i - \bar{y}_i)^2}$$

$$= 1 - \frac{sum\ of\ squared\ residuals}{sum\ of\ total\ squares}$$

- All tell the same story: model is underfit

Lab Time!


lab_class_9_linear_regression_3_and_Cla...
Source on Save
Run
Source
Find
Next
Prev
All
Replace
Replace
All
In selection
Match case
Whole word
Regex
Wrap
#---
Exercises
#---
# 1. Load the movie data set and apply the cleaning code below.
# 2. Set seed of 1818 and Split into testing and training sets
#   using a training proportion of 70%.
# 3. Estimate a model of log movie grossM (Y) on log facebook likes
#   Store this as mod1 and summarize the object.
#   Be sure to use training data to estimate the model!
# 4. Interpret the coefficient on log facebook likes
# 5. Estimate a model of log movie grossM (Y) on log budgetM, and store this
#   as mod2 and summarize
# 6. Interpret the coefficient on log budget M
# 7. Estimate a model of log movie grossM (Y) on log budgetM, limdb_score, and
#   log facebook likes and store this as mod3 and summarize.
# 8. Interpret the coefficient on log of imdb score.
# 9. Upload code to canvas!
library('tidyverse')
library('forcats')
set.seed(1818)
movies <- read.csv(here::here("datasets", "IMDB_movies.csv"))

Class 9: Outline

1. Log Transformations and Interpreting log-log Regressions
2. Model Evaluation
    - Testing and Training Sets, RMSE, and R-Squared
3. Lab Class 9
4. **Why Classification Models?**
5. Logistic Function
6. Log Odds Ratio

Classification examples

Credit Card Default Dataset

Default {ISLR}　　　　R Documentation

Credit Card Default Data

Description

A simulated data set containing information on ten thousand customers. The aim here is to predict which customers will default on their credit card debt.

Usage

`Default`

Format

A data frame with 10000 observations on the following 4 variables.

`default`

A factor with levels `No` and `Yes` indicating whether the customer defaulted on their debt

`student`

A factor with levels `No` and `Yes` indicating whether the customer is a student

`balance`

The average balance that the customer has remaining on their credit card after making their monthly payment

`income`

Income of customer

Source

Simulated data

Why not regression?

```r
library(ISLR)
data(Default)
options(scipen = 3)

library(magrittr)
library(tidyverse)
library(ggExtra)

# create a binary version of default
Default %<>% mutate(default_binary =
                      ifelse(default == "Yes", 1,0))

summary(Default)

# estimate an OLS model using the 0,1
# variable as our dependent variable
mod1 <- lm(default_binary ~ balance,
           data = Default)
summary(mod1)
```

```
> summary(mod1)

Call:
lm(formula = default_binary ~ balance, data = Default)

Residuals:
     Min       1Q   Median       3Q      Max 
-0.23533 -0.06939 -0.02628  0.02004  0.99046 

Coefficients:
               Estimate   Std. Error t value Pr(>|t|)    
(Intercept) -0.075191959  0.003354360  -22.42   <2e-16 ***
balance      0.000129872  0.000003475   37.37   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.1681 on 9998 degrees of freedom
Multiple R-squared:  0.1226,	Adjusted R-squared:  0.1225 
F-statistic:  1397 on 1 and 9998 DF,  p-value: < 2.2e-16
```

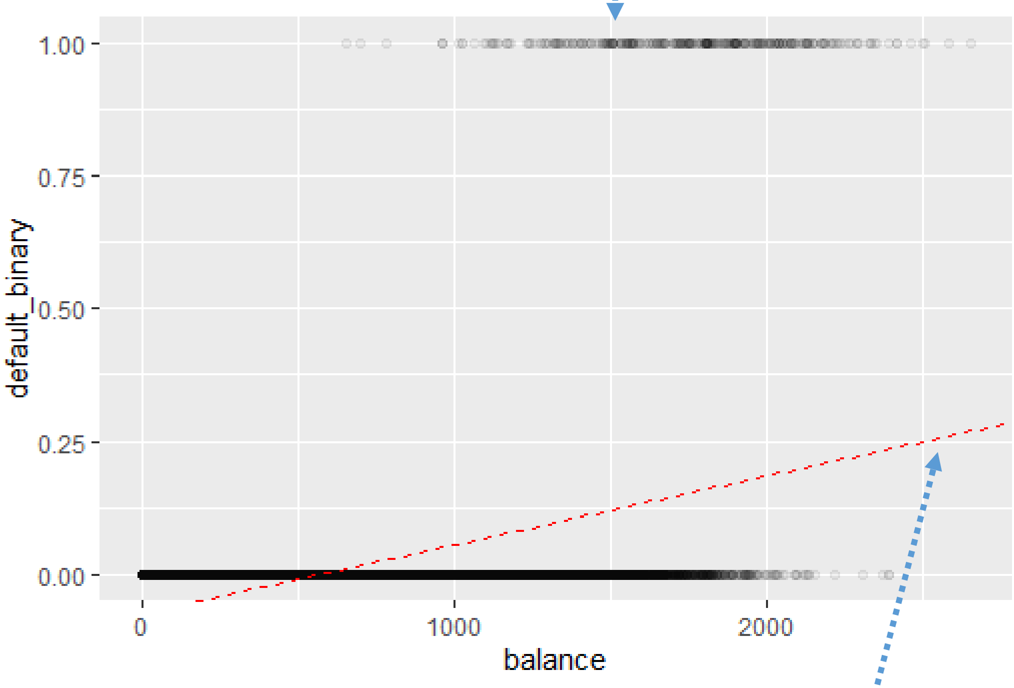
- Let’s estimate a model predicting default based on credit card balance
- R2 looks low but otherwise this model looks perfectly fine

Linear Model to Predict Default

```
preds_DF <- data.frame(
  preds = predict(mod1),
  Default
)

# what kind of predictions do we get for this model?
head(preds_DF)

p <- ggplot(preds_DF, aes(x = balance,
                          y = default_binary)) +
  geom_point(alpha = 1/20) +
  geom_abline(intercept = mod1$coefficients[1],
              slope = mod1$coefficients[2],
              color = "red", linetype = "dashed")

plot(p)
```

- Let’s use the model to generate predictions of default (which is binary)

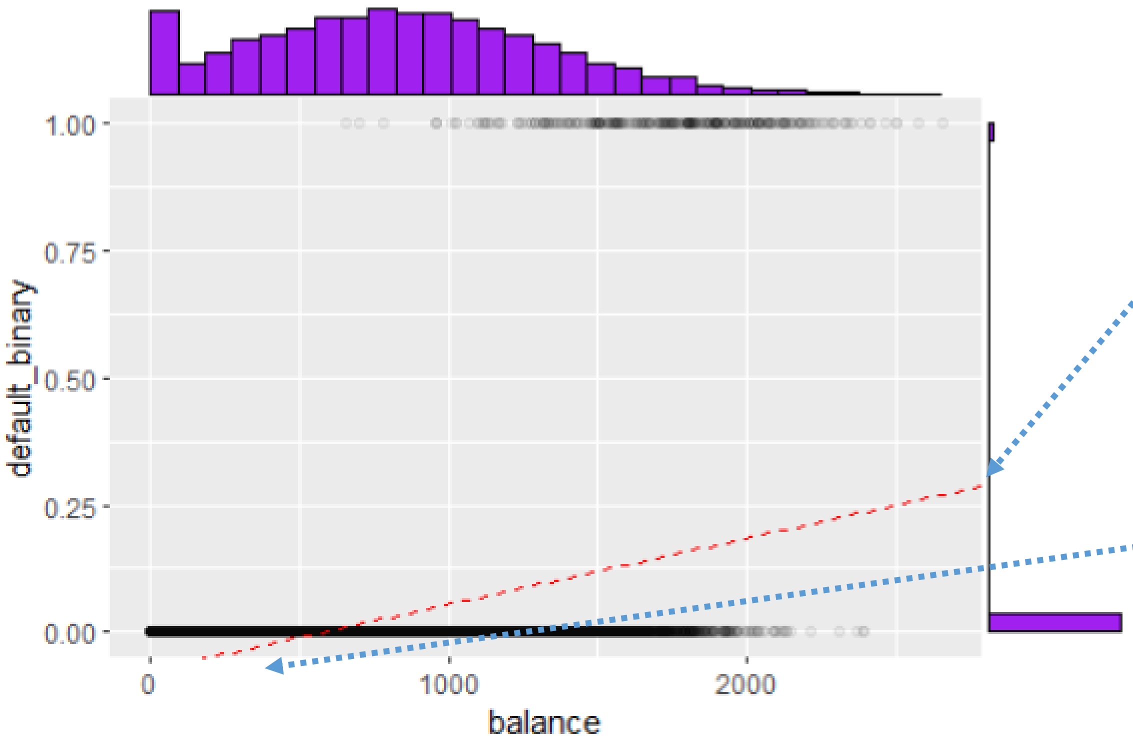
- Black dots show actual default behavior



- Red line shows the predictions from the model

Why is the red line a bad prediction model for default?

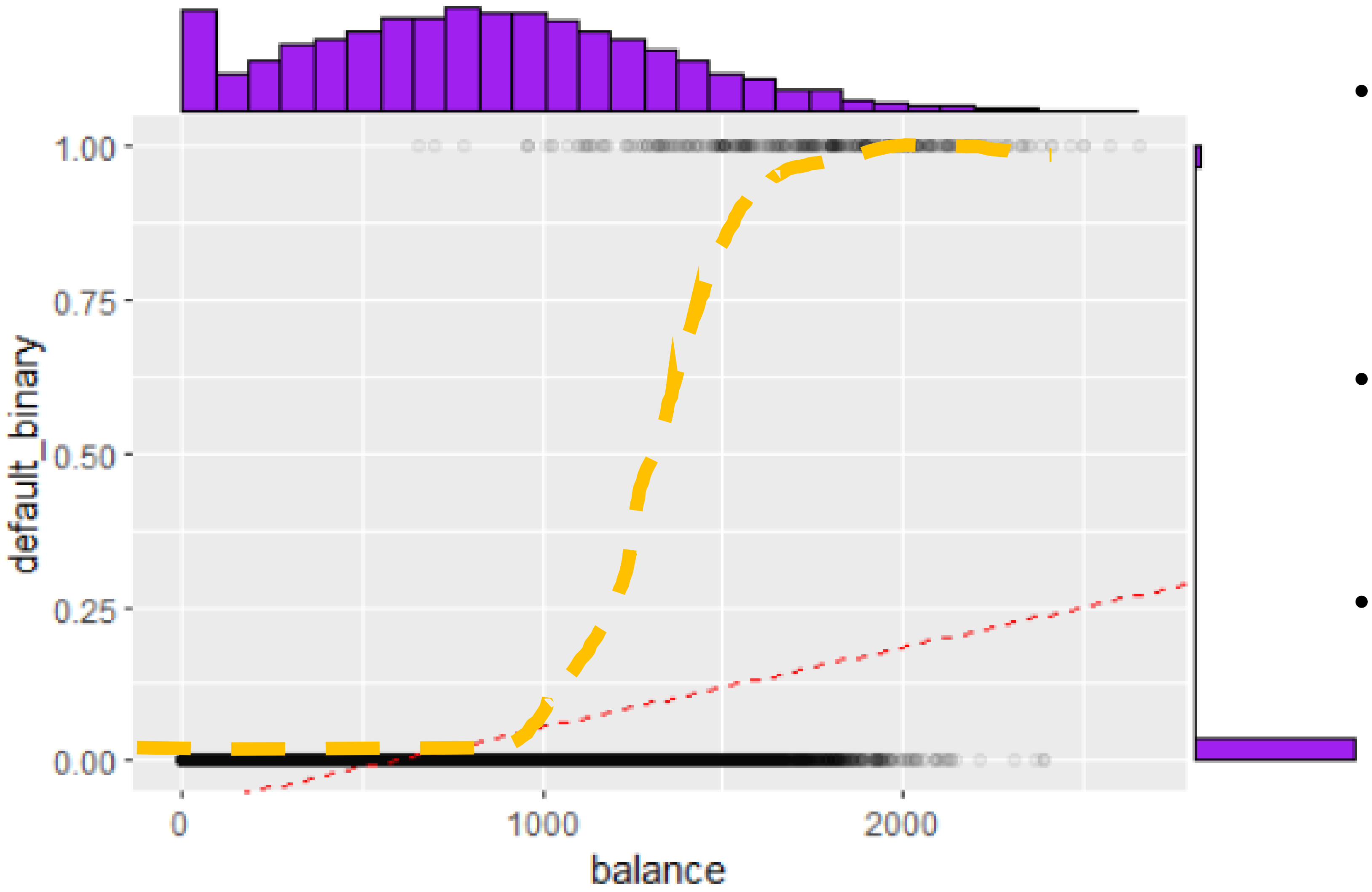
Why Is This a Bad Prediction Model?




```
p <- ggplot(preds_DF, aes(x = balance,
                          y = default_binary)) +
  geom_point(alpha = 1/20) +
  geom_abline(intercept = mod1$coefficients[1],
              slope = mod1$coefficients[2],
              color = "red", linetype = "dashed")

plot(p)
p <- ggMarginal(p, type = "histogram", fill="purple", size=6)
plot(p)
```

1. We never predict more than 30% chance of default!
2. Most observations do not default!
3. We predict negative probability of default!

One Way to Improve the Earlier Model: Squash Predictions



- Because probabilities are between 0 and 1 we want to compress red line to lie within 0 and 1 on the y axis
- i.e let $p(X) = Pr(Y = 1|X)$ be the probability the event occurs
- We want our model to output:

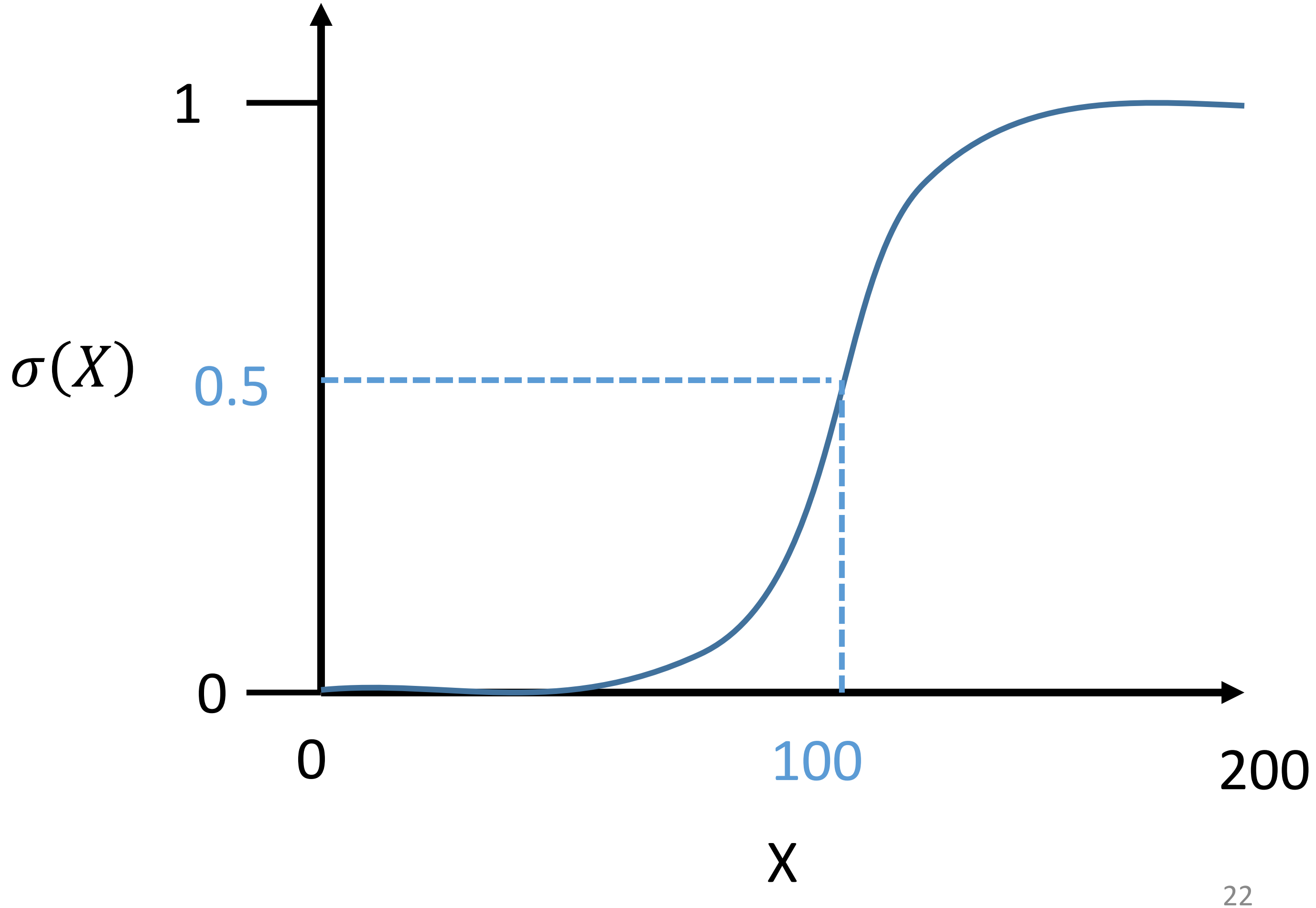
$$Pr(Y = 1|X) \in [0,1]$$

What is The Logistic/Sigmoid Function

- Logisitic is a function that naturally takes inputs X and transforms between 0 and 1

- The logisitic is defined as

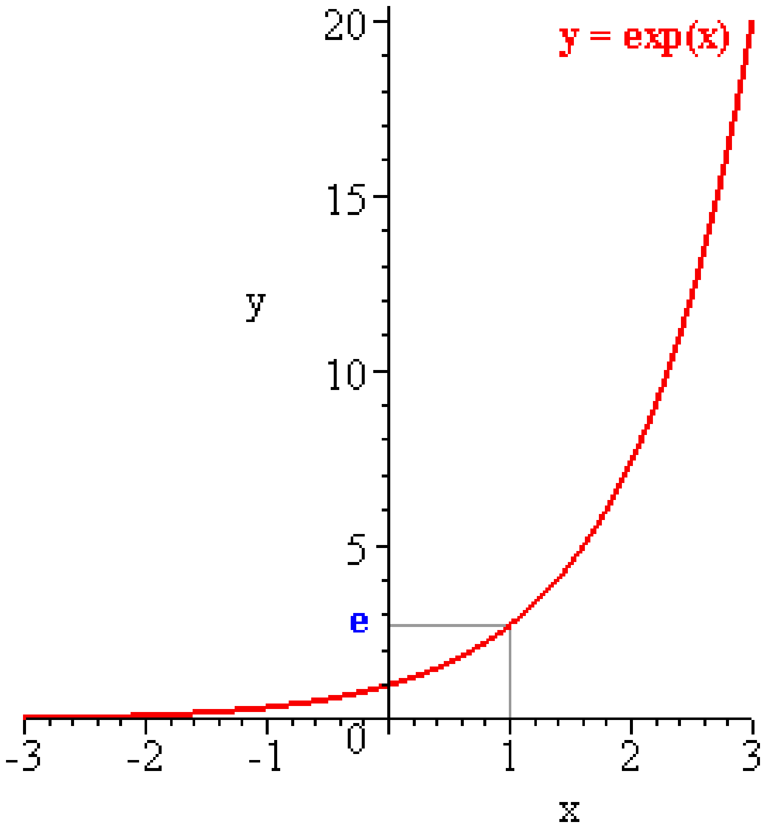
$$\sigma(X) = \frac{1}{1+e^{-X}} = \frac{e^X}{e^X+1}$$



A note on $e^X = \exp(X)$

- Super spooky mathematical function
- $e = 2.718281828459045 \ldots$
- $\frac{d}{dx} e^{\mathrm{X}} = \mathrm{e}^{\mathrm{X}}$ and $e^0 = 1$
    - e.g. rate of increase in function at X is equal to the function at X

Many other ways to characterize function



Using the Logistic/Sigmoid Function to Generate Probabilities

- How do we generate probabilities from this function?

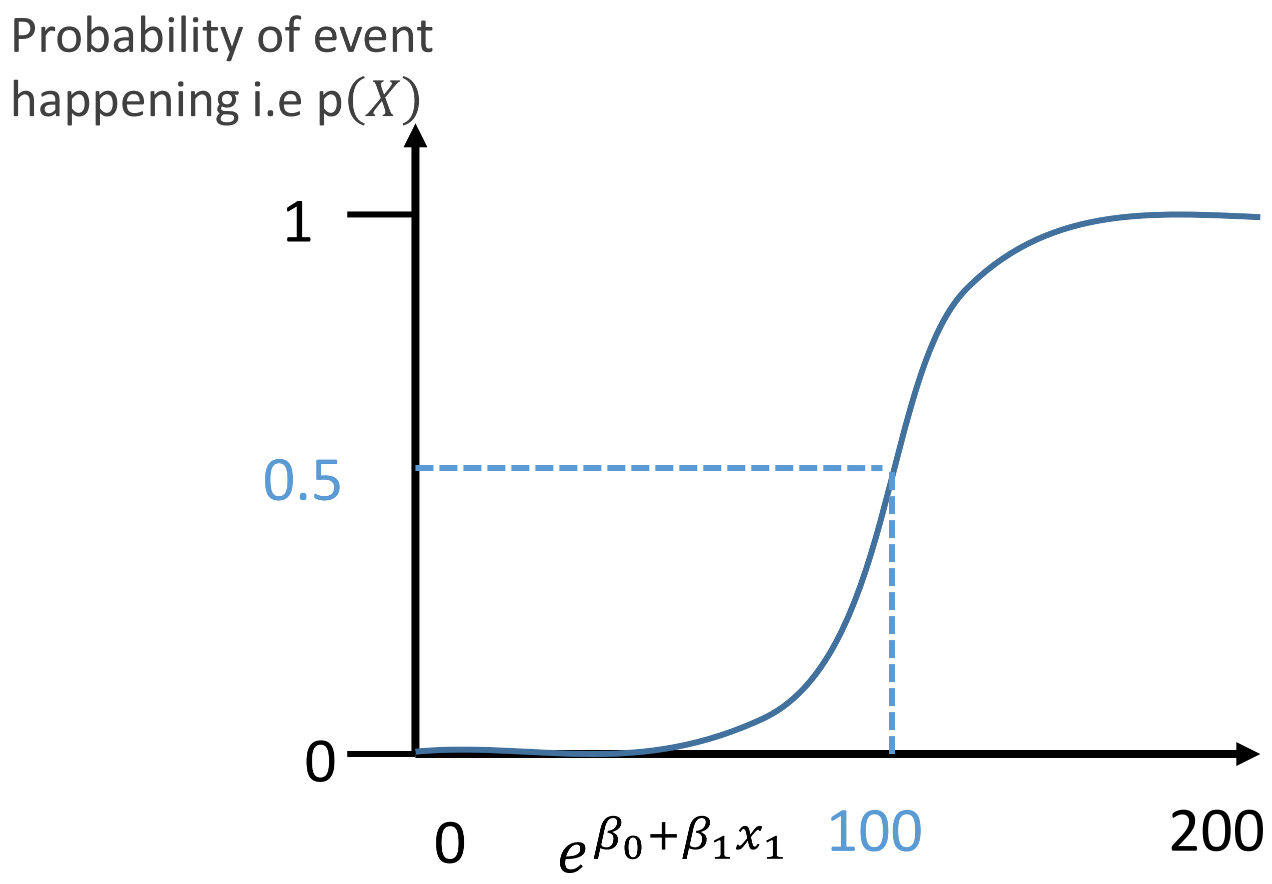
- We let X = $\beta_0 + \beta_1 \cdot x_1 + \cdots + \beta_k \cdot x_k$ and plug this into the logistic function

$$\sigma(X) = \frac{1}{1+e^{-X}} = \frac{e^X}{e^X+1}$$

$$Pr(Y=1|X) = \frac{e^{\beta_0+\beta_1 \cdot x_1}}{e^{\beta_0+\beta_1 \cdot X}+1}$$

$$Pr(Y=1|X) = \frac{1}{1+e^{-(\beta_0+\beta_1 x_1)}}$$

This is equivalent mathematically! I promise. Work it out on pen and paper if you don't believe me



Probability Note on The Complement

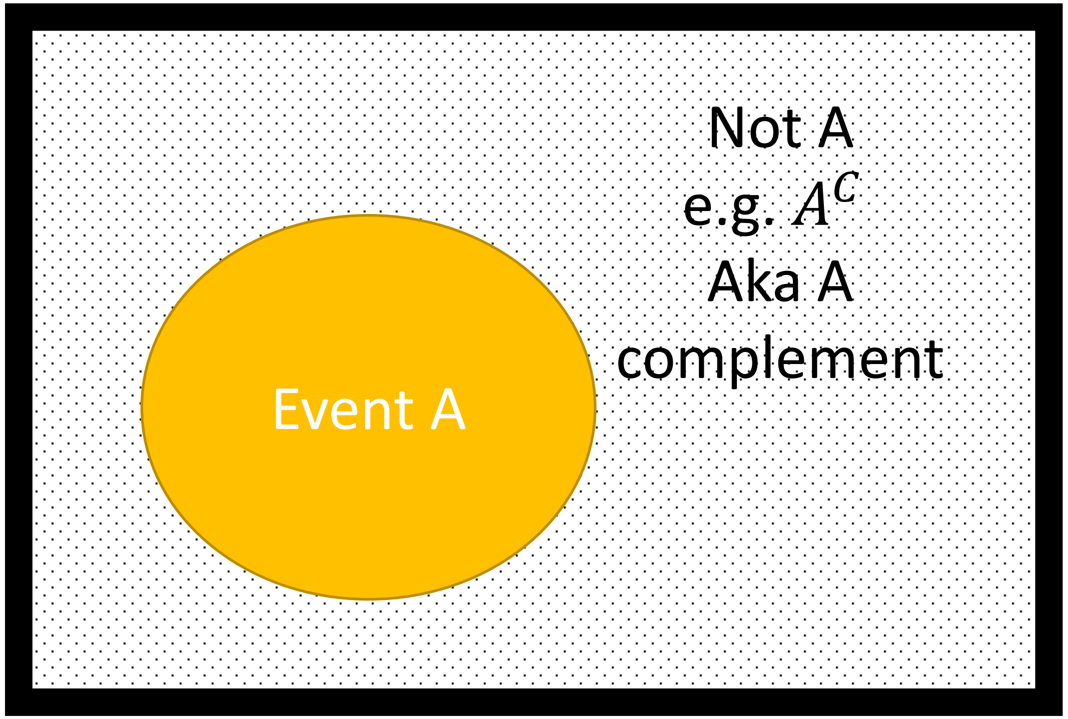
- Q: if $\Pr(A) = 30\%$

- What is the probability of A not happening (the complement) or $\Pr(A^C)$?

- Because events A and not A fully partition the sample space
$$\Pr(A^C) = 1 - P(A)$$

- Fully partition the sample space (i.e. two events are all that can happen):
$$A \cup A^C = \Omega = 1$$

Sample Space (All possible outcomes)



One Weird Trick to Find P(Y=0)

$$Pr(Y = 1|X) = \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

$$Pr(Y = 0|X) = 1 - \Pr(Y = 1|X)$$

$$Pr(Y = 0|X) = 1 - \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

$$Pr(Y = 0|X) = \frac{1 + e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} - \frac{e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}}$$

$$Pr(Y = 0|X) = \frac{1 + e^{\beta_0+\beta_1 X} - e^{\beta_0+\beta_1 X}}{1 + e^{\beta_0+\beta_1 X}} = \frac{1}{1 + e^{\beta_0+\beta_1 X}}$$

Expressing Ratio of Probabilities: Odds Ratio

- Armed with P(Y=1) and P(Y=0) we know probabilities for each of these events
- A useful expression is the odds ratio, or the ratio of events occurring

$$\frac{p(Y=1|X)}{p(Y=0|\,X)} =$$

$$= \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \Big/ \frac{1}{1+e^{\beta_0+\beta_1 X}}$$

$$= \frac{e^{\beta_0+\beta_1 X}}{1+e^{\beta_0+\beta_1 X}} \cdot \frac{1+e^{\beta_0+\beta_1 X}}{1}$$

$$= e^{\beta_0+\beta_1 X}$$

Example odds ratio: pr(Y = “rain” | X)

$$\frac{p(Y = rain|X)}{1 - p(Y = rain|X)} =$$

$$= \frac{0.2}{0.8} = \frac{1}{4}$$

Boston, MA 10 Day Weather

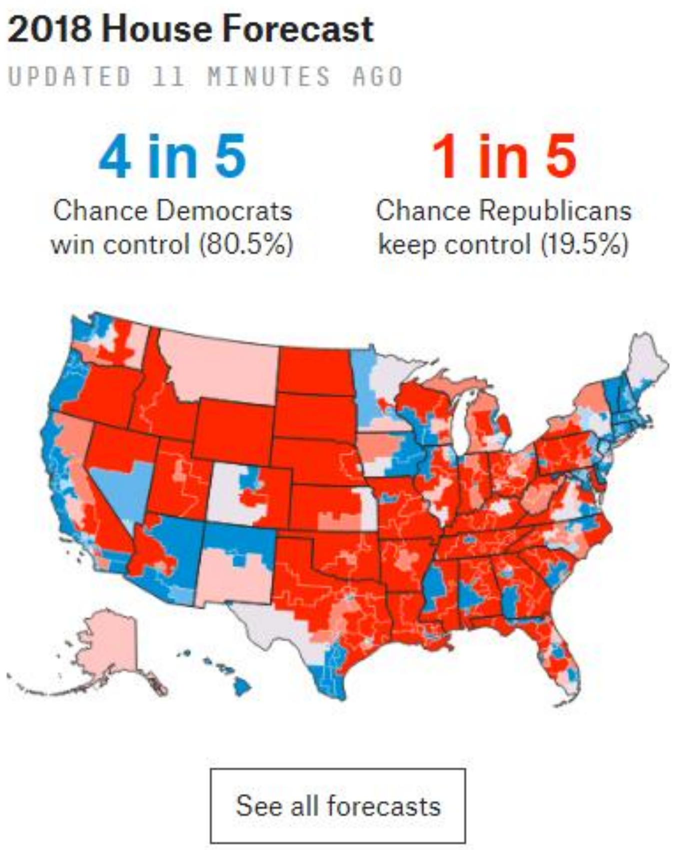
12:31 pm EDT12:29 pm EDT Print

DAY	DESCRIPTION	HIGH / LOW	PRECIP
TODAY SEP 19	Cloudy	64°/58°	0%
THU SEP 20	Partly Cloudy	65°/58°	20%

Example odds ratio: pr(“Dems win house”|X)

$$\frac{p(Y = \text{“}Dems\ win\text{”}|X)}{1 - p(Y = \text{“}Dems\ win\text{”}|X)} =$$

$$= \frac{0.8}{0.2} = 4$$



Logit Models Model the Outcome As a Log Odds Ratio

$$\frac{p(Y=1|X)}{p(Y=0|\ X)} = e^{\beta_0+\beta_1 X}$$

The outcome variable (Y) for a logistic regression is the log odds ratio

$$log\left(\frac{p(Y=1|X)}{p(Y=0|\ X)}\right) = \log\left(e^{\beta_0+\beta_1 X}\right)$$

Log odds ratio is a linear expression of constants and coefficients of a nonlinear process!

$$log\left(\frac{p(Y=1|X)}{p(Y=0|\ X)}\right) = \beta_0 + \beta_1 X$$

All logistic coefficients can be interpreted as impact on log odds ratio

Class 9 Summary

- Log transformations of Y or X variables is useful when the data are “spread out”
- We interpret log-log regression coefficients as elasticities: a 1% change in the X variable leads to a coefficient % change in the Y variable
- We split data into testing and training sets, estimate a model on the training set and evaluate on the test set
- Logit functions compress predictions to lie between 0 and 1, which are valid probabilities
- The logistic model models the outcome (Y) as the log odds ratio!