STA 235H - Model Selection II: Shrinkage

Fall 2022

McCombs School of Business, UT Austin

Last class

- Started with our prediction chapter
  - Bias vs. Variance
  - Validation set approach and Cross-validation
  - How to choose a model for a continuous outcome (RMSE)
  - Stepwise selection

Knowledge check from last week

1) Which model is higher bias: A complex model or a simpler one?

2) Why do we split our data into training and testing datasets?

3) How do we compare models with continuous outcomes?

How forward stepwise selection works: Example from last class

1) Start with a null model (no covariates)

- Your best guess will be the average of the outcome in the training dataset!

2) Test out all models with **one covariate**, and **select the best one**:

- E.g. $logins \sim female$, $logins \sim succession$, $logins \sim age$, ...
- $logins \sim succession$ is the best one (according to RMSE)

3) Test out all models with **two covariates**, but that have $succession$!

- E.g. $logins \sim succession + female$, $logins \sim succession + age$, $logins \sim succession + city$, ...

4) You will end up with k possible models (k: total number of predictors).

- Choose the best one, depending on the RMSE.

Today: Continuing our journey

- How to improve our linear regressions:
  - Ridge regression
  - Lasso regression
- Look at binary outcomes

Honey, I shrunk the coefficients!

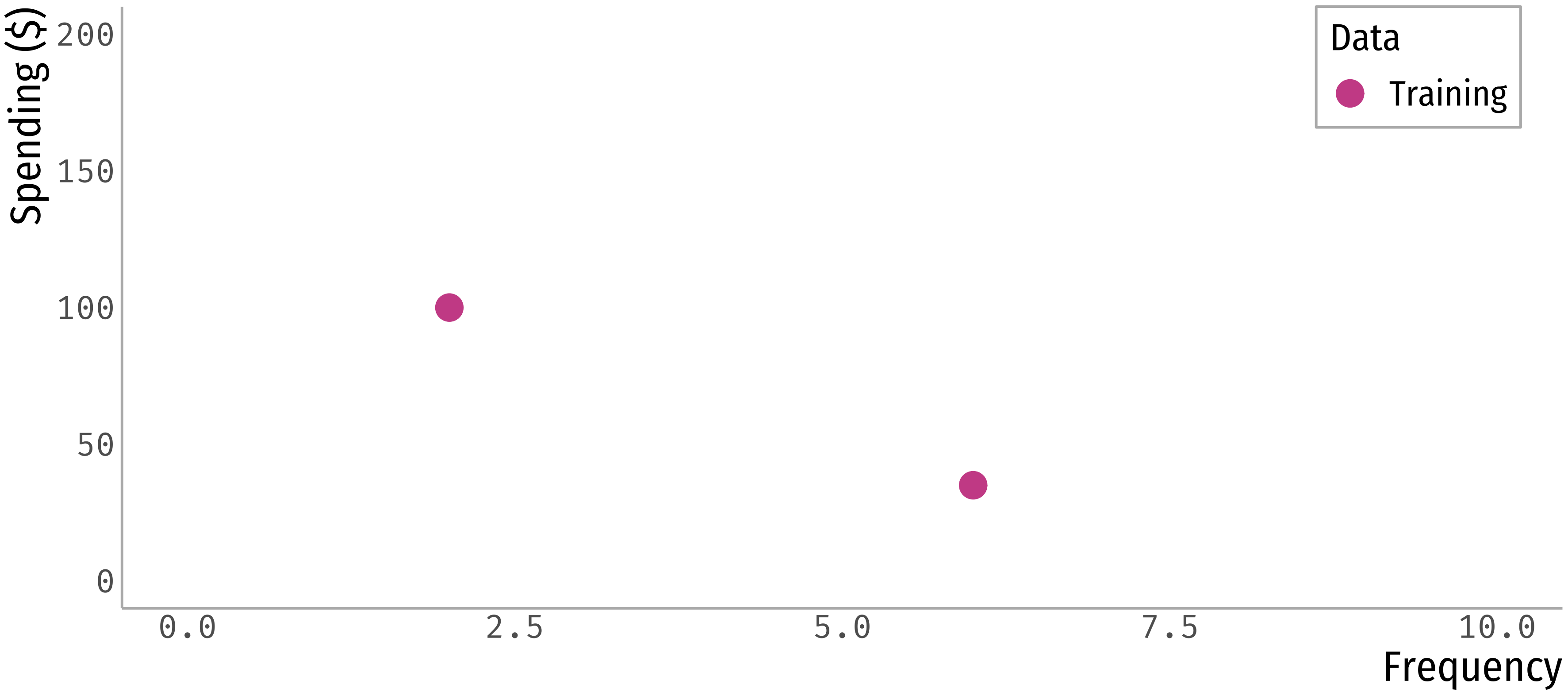
What is shrinkage?

- We reviewed the **stepwise procedure**: Subsetting model selection approach.
  - Select k out of p total predictors
- **Shrinkage *(a.k.a Regularization)***: Fitting a model with all p predictors, but introducing bias (i.e. shrinking coefficients towards 0) for improvement in variance.
  - Ridge regression
  - Lasso regression

On top of a ridge.

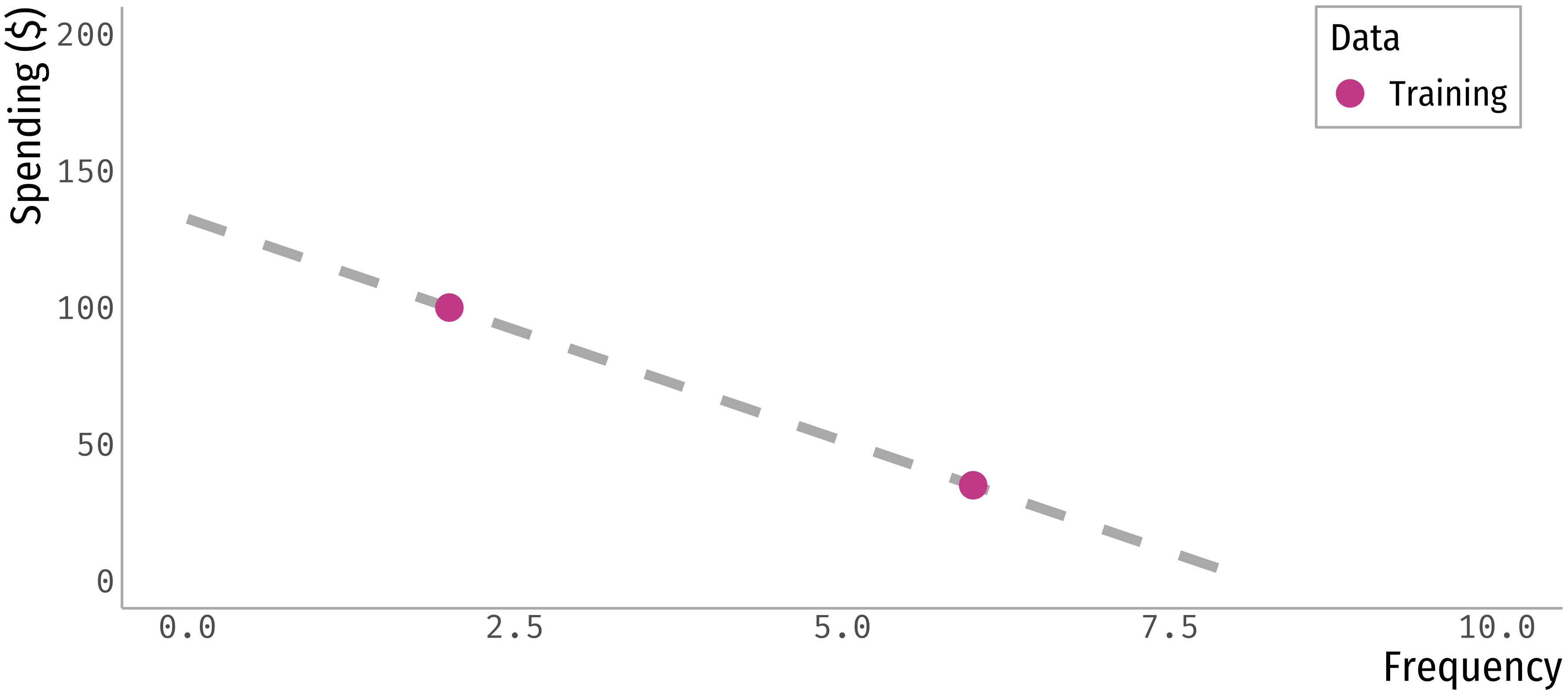
Ridge Regression: An example

- Predict spending based on frequency of visits to a website



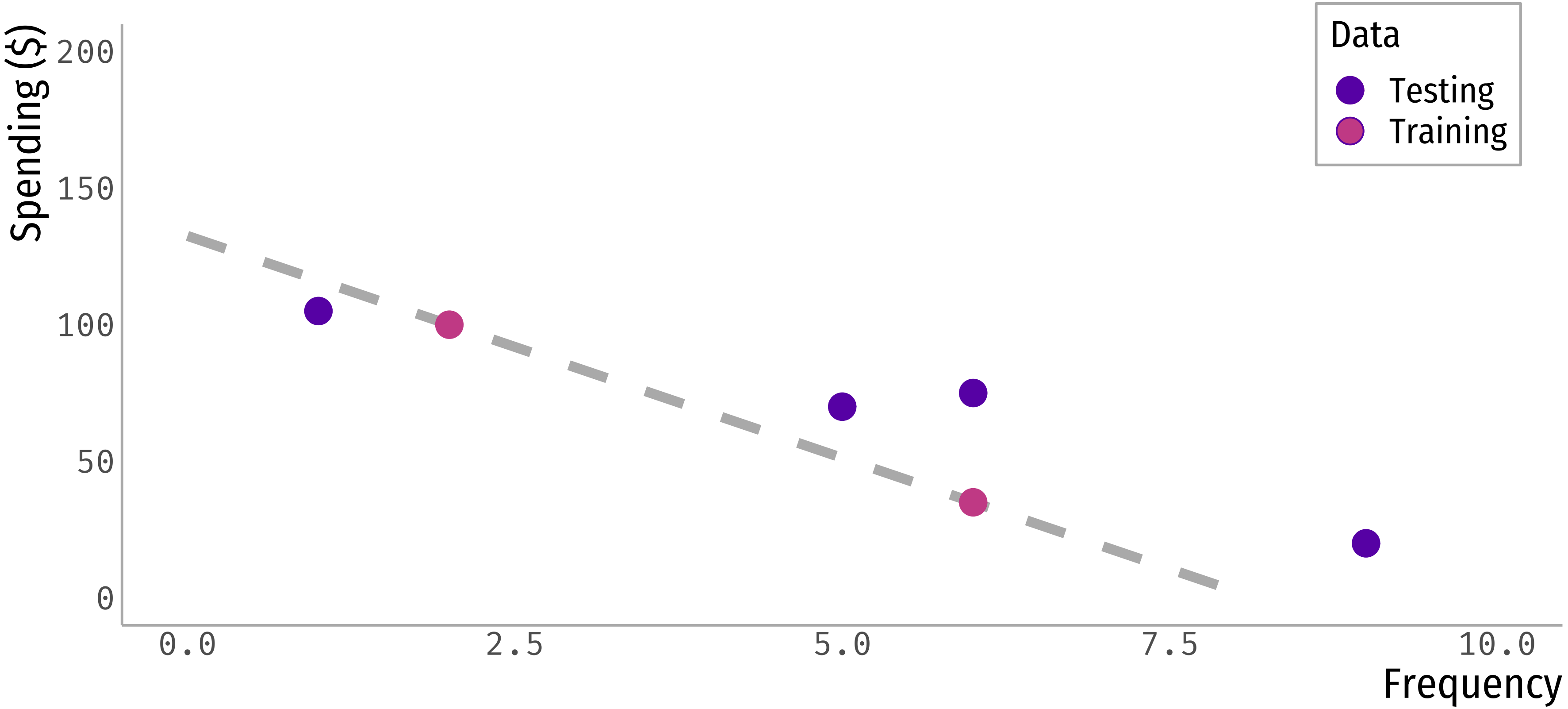
Ordinary Least Squares

- In an OLS: Minimize sum of squared-errors, i.e. $\min_{\beta} \sum_{i=1}^{n} (\text{spend}_i - (\beta_0 + \beta_1 \text{freq}_i))^2$



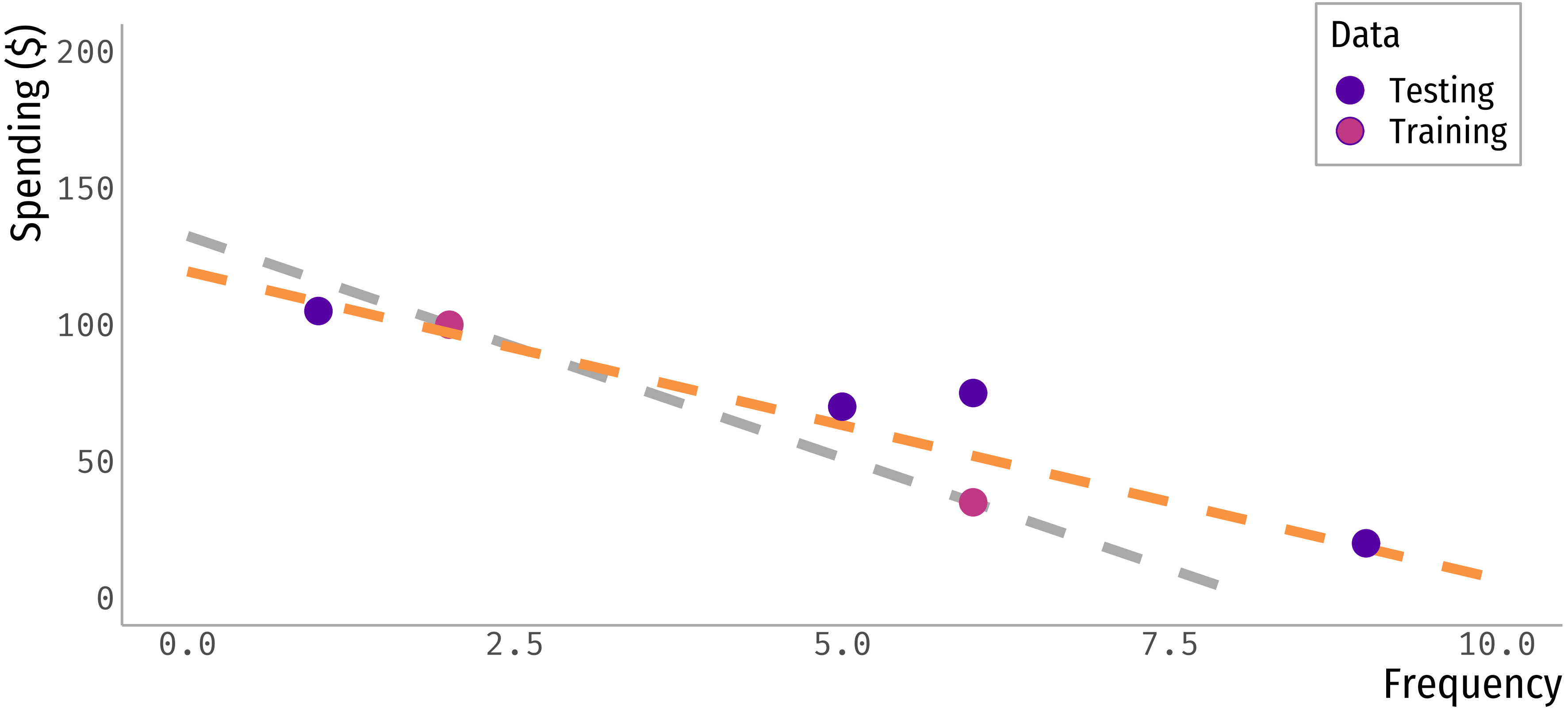
What about fit?

- Does the OLS fit the testing data well?



Ridge Regression

- Let's shrink the coefficients!: Ridge Regression



Ridge Regression: What does it do?

- Ridge regression **introduces bias to reduce variance** in the testing data set.

- In a simple regression (i.e. one regressor/covariate):

$$\min_{\beta} \sum_{i=1}^{n} \underbrace{(y_i - \beta_0 - x_i\beta_1)^2}_{OLS}$$

Ridge Regression: What does it do?

- Ridge regression **introduces bias to reduce variance** in the testing data set.
- In a simple regression (i.e. one regressor/covariate):

$$\min_{\beta} \sum_{i=1}^{n} \underbrace{(y_i - \beta_0 - x_i\beta_1)^2}_{OLS} + \underbrace{\lambda \cdot \beta_1^2}_{Ridge Penalty}$$

- λ is the **penalty factor** $\rightarrow$ indicates how much we want to shrink the coefficients.

Q1: In general, which model will have smaller β coefficients?

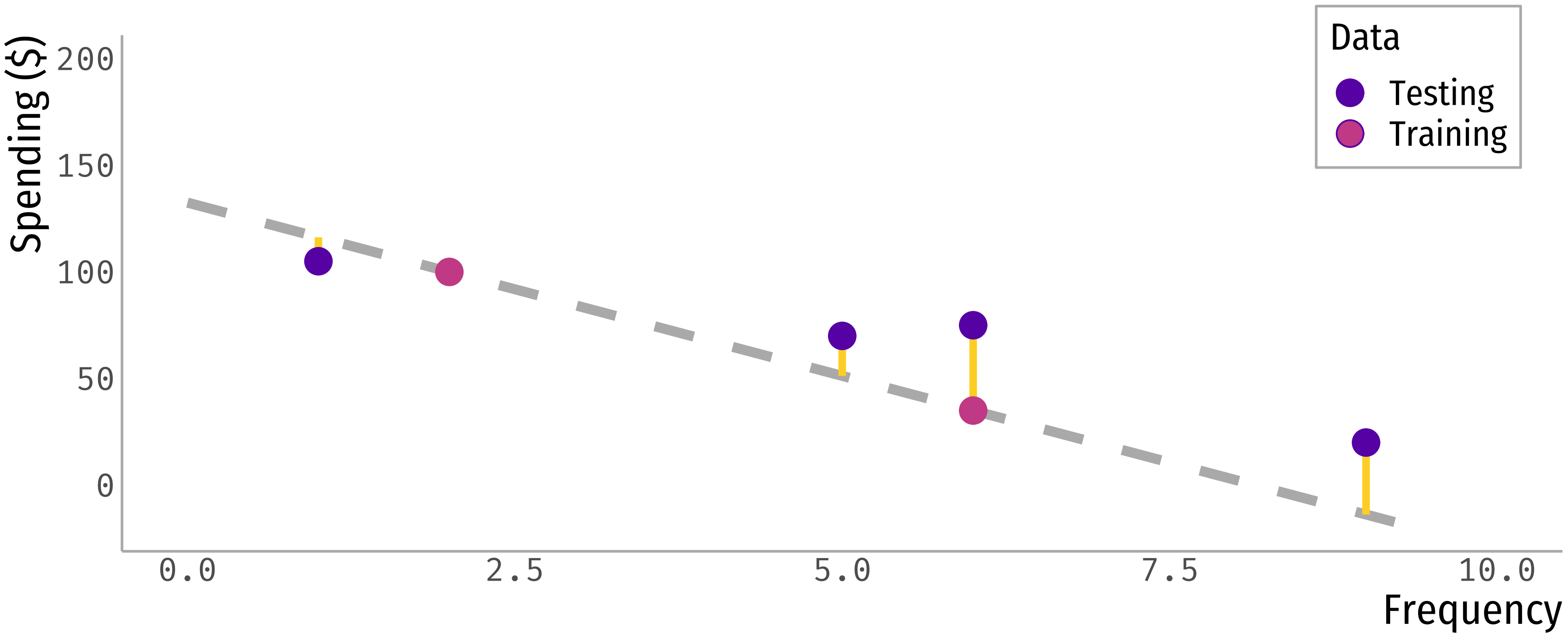
a) A model with a larger λ

b) A model with a smaller λ

Remember… we care about accuracy in the testing dataset!

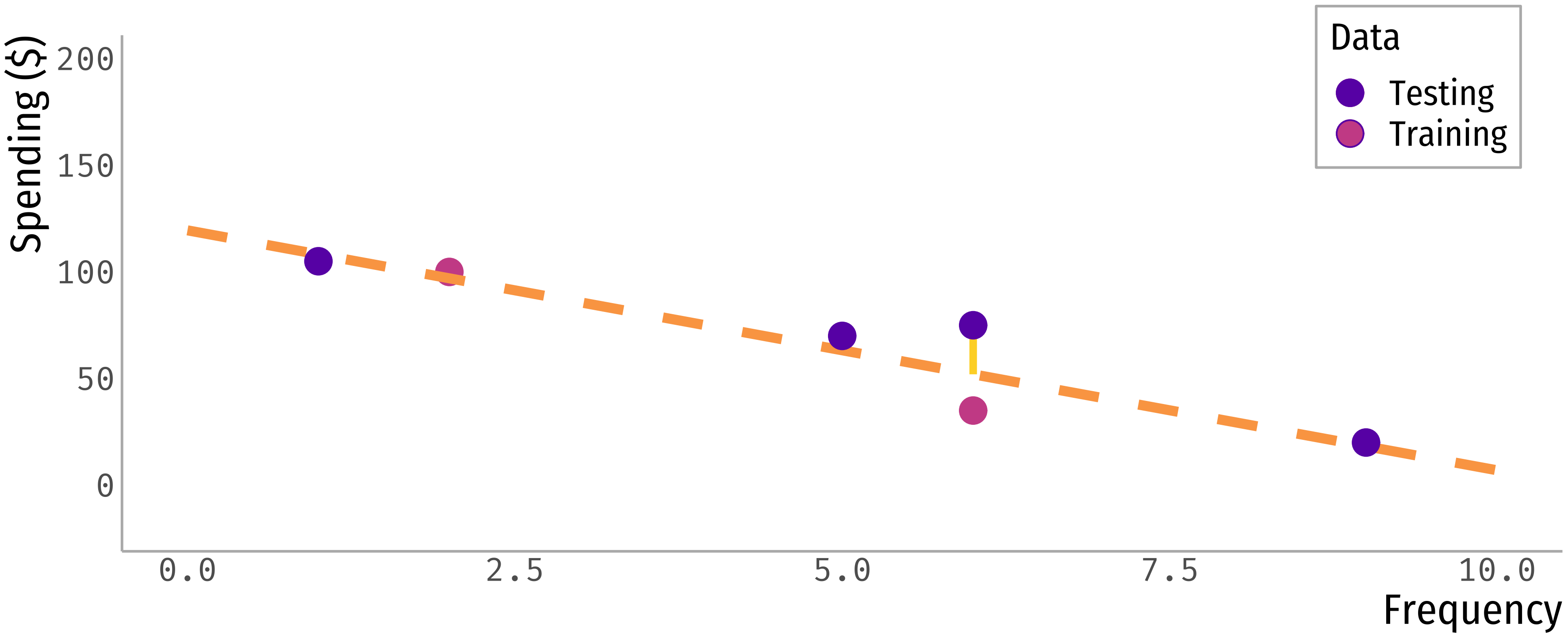
RMSE on the testing dataset: OLS

$$RMSE = \sqrt{\frac{1}{4}\sum_{i=1}^{4}(\text{spend}_i - (132.5 - 16.25 \cdot \text{freq}_i))^2} = 28.36$$



RMSE on the testing dataset: Ridge Regression

$$RMSE = \sqrt{\frac{1}{4}\sum_{i=1}^{4}(\text{spend}_i - (119.5 - 11.25 \cdot \text{freq}_i))^2} = 12.13$$



Ridge Regression in general

- For regressions that include **more than one regressor**:

$$\min_{\beta} \sum_{i=1}^{n} \underbrace{(y_i - \sum_{k=0}^{p} x_i \beta_k)^2}_{OLS} + \underbrace{\lambda \cdot \sum_{k=1}^{p} \beta_k^2}_{Ridge Penalty}$$

- In our previous example, if we had two regressors, $female$ and $freq$:

$$\min_{\beta} \sum_{i=1}^{n} (\text{spend}_i - \beta_0 - \beta_1 \text{female}_i - \beta_2 \text{freq}_i)^2 + \lambda \cdot (\beta_1^2 + \beta_2^2)$$

- Because the ridge penalty includes the β's coefficients, **scale matters**:
  - Standardize variables (*you will do that as an option in your code*)

How do we choose λ?

Cross-validation!

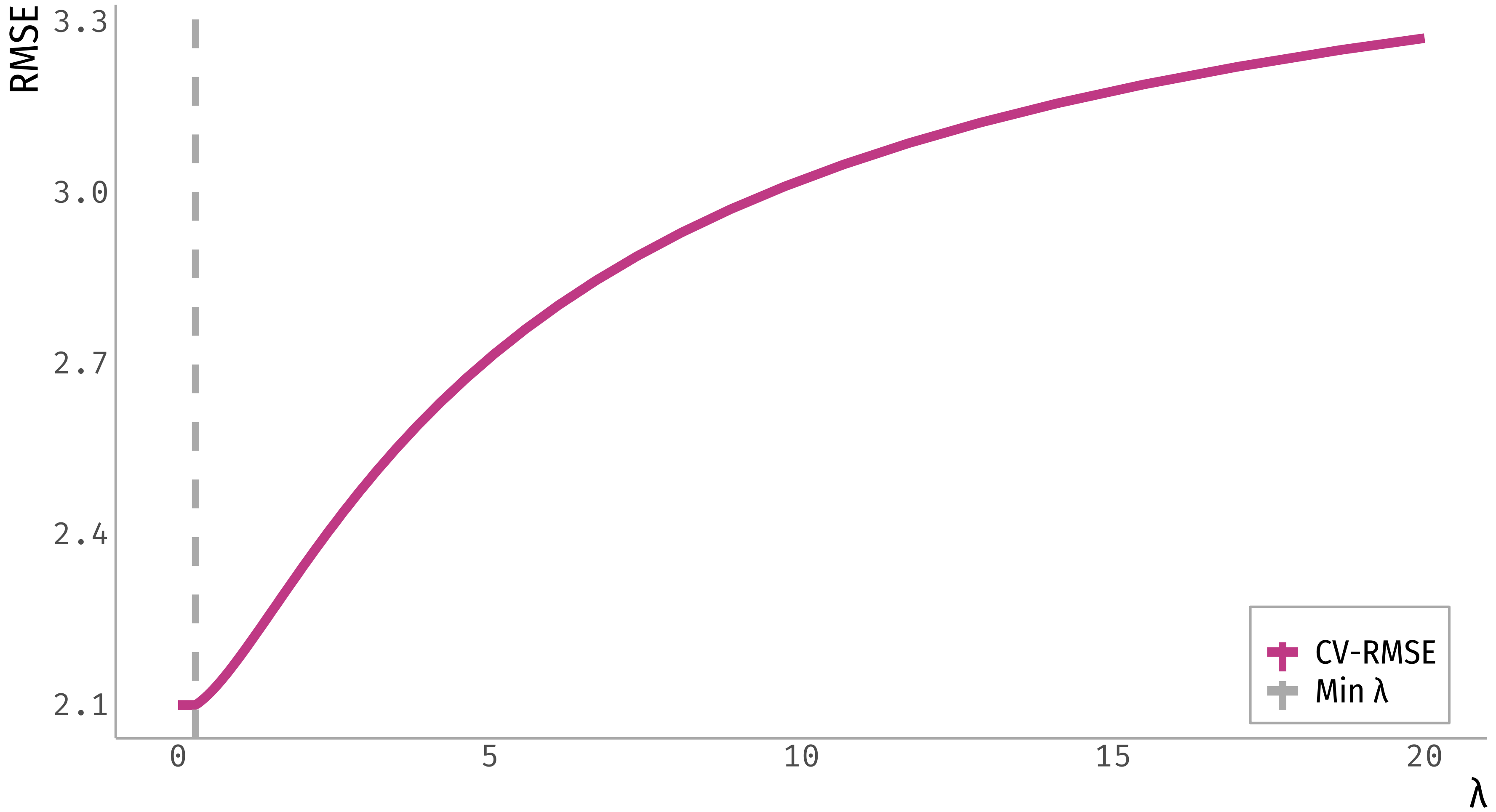
1) Choose a grid of λ values

- The grid you choose will be context dependent (play around with it!)

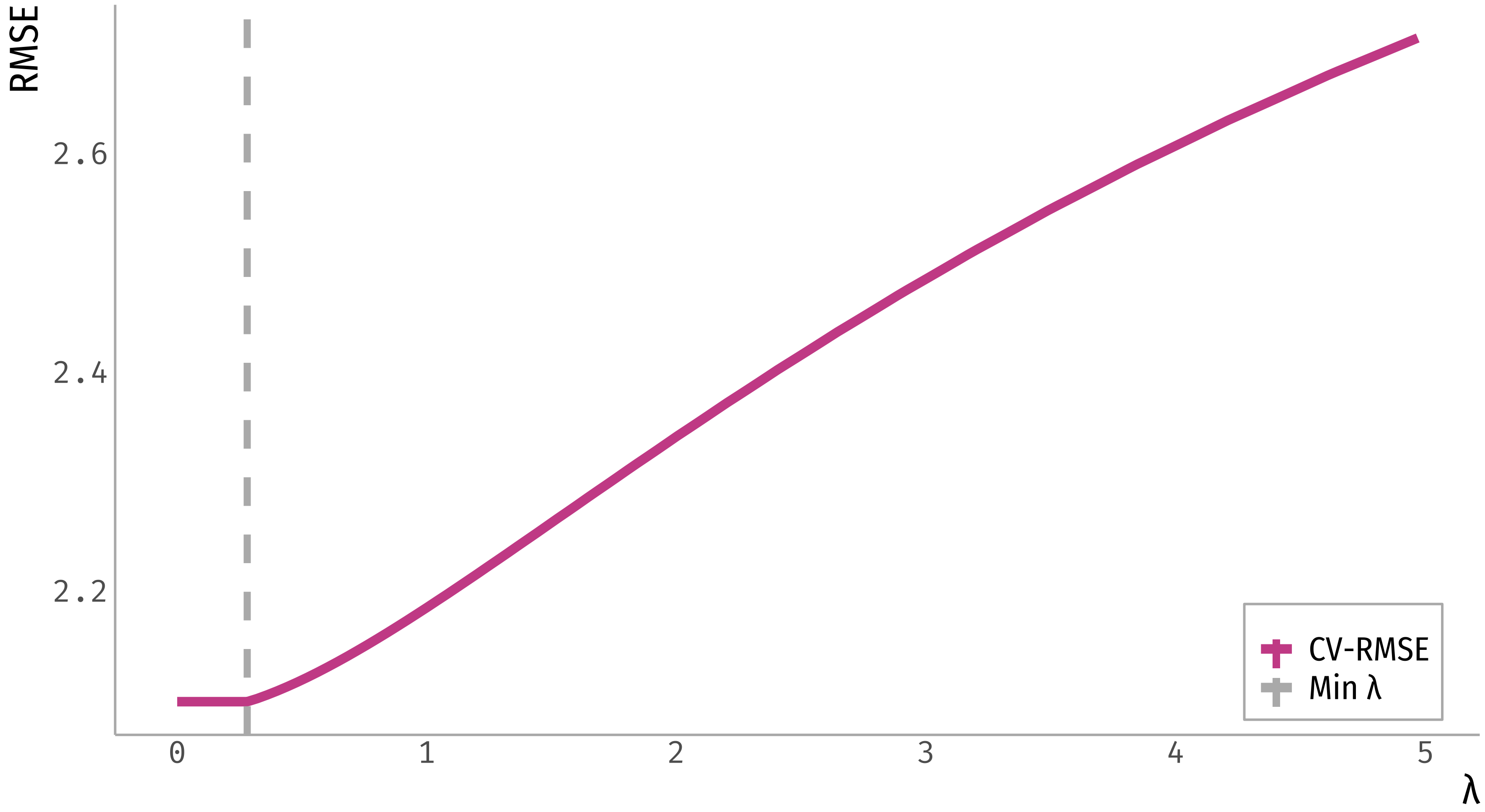
2) Compute cross-validation error (e.g. RMSE) for each

3) Choose the smallest one.

λ vs RMSE?



λ vs RMSE? A zoom



How do we do this in R?

```r
library(caret)

set.seed(100)

hbo = read.csv("https://raw.githubusercontent.c

lambda_seq = seq(0, 20, length = 500)

ridge = train(logins ~ . - unsubscribe - id,
            data = train.data,
            method = "glmnet",
            preProcess = "scale",
            trControl = trainControl("cv", numb
            tuneGrid = expand.grid(alpha = 0,
                          lambda = lambda_seq)
  )

cv_lambda = data.frame(lambda = ridge$results$l
                       rmse = ridge$results$RM
```

- We will be using the `caret` package

How do we do this in R?

```r
library(caret)

set.seed(100)

hbo = read.csv("https://raw.githubusercontent.c

lambda_seq = seq(0, 20, length = 500)

ridge = train(logins ~ . - unsubscribe - id,
            data = train.data,
            method = "glmnet",
            preProcess = "scale",
            trControl = trainControl("cv", numb
            tuneGrid = expand.grid(alpha = 0,
                                   lambda = lambda_seq)
  )
```

- We will be using the `caret` package
- We are doing **cross-validation**, so remember to set a seed!

How do we do this in R?

```r
library(caret)

set.seed(100)

hbo = read.csv("https://raw.githubusercontent.c

lambda_seq = seq(0, 20, length = 500)

ridge = train(logins ~ . - unsubscribe - id,
             data = train.data,
             method = "glmnet",
             preProcess = "scale",
             trControl = trainControl("cv", numb
             tuneGrid = expand.grid(alpha = 0,
                                    lambda = lambda_seq)
  )

cv_lambda = data.frame(lambda = ridge$results$l
                                rmse = ridge$results$RM
```

- We will be using the `caret` package
- We are doing **cross-validation**, so remember to set a seed!
- You need to create a grid for the λ's **that will be tested**

How do we do this in R?

```r
library(caret)

set.seed(100)

hbo = read.csv("https://raw.githubusercontent.c

lambda_seq = seq(0, 20, length = 500)

ridge = train(logins ~ . - unsubscribe - id,
              data = train.data,
              method = "glmnet",
              preProcess = "scale",
              trControl = trainControl("cv", numb
              tuneGrid = expand.grid(alpha = 0,
                                     lambda = lambda_seq)
   )

cv_lambda = data.frame(lambda = ridge$results$l
                       rmse = ridge$results$RM
```

- We will be using the `caret` package
- We are doing **cross-validation**, so remember to set a seed!
- You need to create a grid for the λ's **that will be tested**
- The function we will use is `train`: Same as before
    - `method="glmnet"` means that it will run an elastic net.
    - `alpha=0` means is a **ridge regression**
    - `lambda = lambda_seq` is not necessary (you can provide your own grid)

How do we do this in R?

```
library(caret)

set.seed(100)

hbo = read.csv("https://raw.githubusercontent.c

lambda_seq = seq(0, 20, length = 500)

ridge = train(logins ~ . - unsubscribe - id,
            data = train.data,
            method = "glmnet",
            preProcess = "scale",
            trControl = trainControl("cv", numb
            tuneGrid = expand.grid(alpha = 0,
                                   lambda = lambda_seq)
  )

cv_lambda = data.frame(lambda = ridge$results$l
                       rmse = ridge$results$RM
```

- We will be using the `caret` package
- We are doing **cross-validation**, so remember to set a seed!
- You need to create a grid for the λ's **that will be tested**
- The function we will use is `train`: Same as before
- Important objects in `cv`:
    - `results$lambda`: Vector of λ that was tested
    - `results$RMSE`: RMSE for each λ
    - `bestTune$lambda`: λ that minimizes the error term.

How do we do this in R?

OLS regression:

```
lm1 = lm(logins ~ succession + city,
          data = train.data)

coef(lm1)
```

```
## (Intercept)  succession        city
##    7.035888   -6.306371    2.570454
```

```
rmse(lm1, test.data)
```

```
## [1] 2.089868
```

Ridge regression:

```
coef(ridge$finalModel, ridge$bestTune$lambda)
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  6.564243424
## female       0.002726465
## city         0.824387472
## age          0.046468790
## succession  -2.639308962
```

```
rmse(ridge, test.data)
```

```
## [1] 2.097452
```

Throwing a lasso

Lasso regression

- Very similar to ridge regression, except it **changes the penalty term**:

$$\min_{\beta} \sum_{i=1}^{n} \underbrace{(y_i - \sum_{k=0}^{p} x_i \beta_k)^2}_{OLS} + \underbrace{\lambda \cdot \sum_{k=1}^{p} |\beta_k|}_{LassoPenalty}$$

- In our previous example:

$$\min_{\beta} \sum_{i=1}^{n} (\text{spend}_i - \beta_0 - \beta_1 \text{female}_i - \beta_2 \text{freq}_i)^2 + \lambda \cdot (|\beta_1| + |\beta_2|)$$

- Lasso regression is also called l_1 regularization:

$$||\beta||_1 = \sum_{k=1}^{p} |\beta|$$

Ridge vs Lasso

Ridge

- Final model will have p coefficients
- Usually better with multicollinearity

Lasso

- Can set coefficients = 0
- Improves interpretability of model
- Can be used for model selection

And how do we do Lasso in R?

```
library(caret)

set.seed(100)

hbo = read.csv("https://raw.githubusercontent.c

lambda_seq = seq(0, 20, length = 500)

lasso = train(logins ~ . - unsubscribe - id, da
            method = "glmnet",
            preProcess = "scale",
            trControl = trainControl("cv", numb
            tuneGrid = expand.grid(alpha = 1,
                                   lambda = lambda_seq)
  )

cv_lambda = data.frame(lambda = lasso$results$l
                       rmse = lasso$results$RM
```

Exactly the same!

- … But change `alpha=1`!!

And how do we do Lasso in R?

Ridge regression:

```
coef(ridge$finalModel, ridge$bestTune$lambda)
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                         s1
## (Intercept)  6.564243424
## female       0.002726465
## city         0.824387472
## age          0.046468790
## succession  -2.639308962
```

```
rmse(ridge, test.data)
```

```
## [1] 2.097452
```

Lasso regression:

```
coef(lasso$finalModel, lasso$bestTune$lambda)
```

```
## 5 x 1 sparse Matrix of class "dgCMatrix"
##                        s1
## (Intercept)  6.84122778
## female       .
## city         0.87982819
## age          0.03099797
## succession  -2.83492585
```

```
rmse(lasso, test.data)
```

```
## [1] 2.09171
```

A note on binary outcomes

- If we are predicting binary outcomes, RMSE would not be an appropriate measure anymore!
  - We will use accuracy instead: The proportion (%) of correctly classified observations.
- For example:

```
set.seed(100)

lasso = train(factor(unsubscribe) ~ . - id, data = train.data,
              method = "glmnet", preProcess = "scale",
              trControl = trainControl("cv", number = 10),
              tuneGrid = expand.grid(alpha = 1, lambda = lambda_seq))

pred.values = lasso %>% predict(test.data)

mean(pred.values == test.data$unsubscribe)
```

```
## [1] 0.736
```

A note on binary outcomes

- If we are predicting binary outcomes, RMSE would not be an appropriate measure anymore!
  - We will use accuracy instead: The proportion (%) of correctly classified observations.
- For example:

```
set.seed(100)

lasso = train(factor(unsubscribe) ~ . - id, data = train.data,
            method = "glmnet", preProcess = "scale",
            trControl = trainControl("cv", number = 10),
            tuneGrid = expand.grid(alpha = 1, lambda = lambda_seq))

pred.values = lasso %>% predict(test.data)

mean(pred.values == test.data$unsubscribe)
```

```
## [1] 0.736
```

A note on binary outcomes

- If we are predicting **binary outcomes**, RMSE would not be an appropriate measure anymore!
  - We will use **accuracy instead**: The proportion (%) of correctly classified observations.
- For example:

```
set.seed(100)

lasso = train(factor(unsubscribe) ~ . - id, data = train.data,
            method = "glmnet", preProcess = "scale",
            trControl = trainControl("cv", number = 10),
            tuneGrid = expand.grid(alpha = 1, lambda = lambda_seq))

pred.values = lasso %>% predict(test.data)

mean(pred.values == test.data$unsubscribe)
```

```
## [1] 0.736
```

Your Turn

Main takeway points

- You can **shrink coefficients** to introduce bias and decrease variance.
- Ridge and Lasso regression are **similar**:
  - Lasso can be used for model selection.
- Importance of understanding **how to estimate the penalty coefficient**.

References

- James, G. et al. (2021). "Introduction to Statistical Learning with Applications in R". *Springer. Chapter 6.*
- STDHA. (2018). "Penalized Regression Essentials: Ridge, Lasso & Elastic Net"