# STAT 215A Fall 2020 Week 10

James Duncan, OH: M, Th 2-4pm

Thanks to Tiffany Tang for sharing her slides

## Announcements

- Congrats on completing the midterm!
- Lab 4 will be released Monday
  - 18 days instead of 14 to complete, due 11/19 at 11:59pm
  - Reach out to your group members!
- Final lab pushed back one week
  - Released 11/20
  - Due 12/11 at 11:59pm
- Next week's plan
  - No pre-lecture this week
  - Bin will talk about exponential family, GLMs, and logistic regression and then go on to inference
  - Take a look at Chs. 3, 5, & 7 in the Dobson book (on bCourses)

## Outline for today

- Lab 2 review
- Lab 3 post-mortem
- Regularization Pt. 2
- Data splitting and statistical learning pipeline
- Introduction to Lab 4

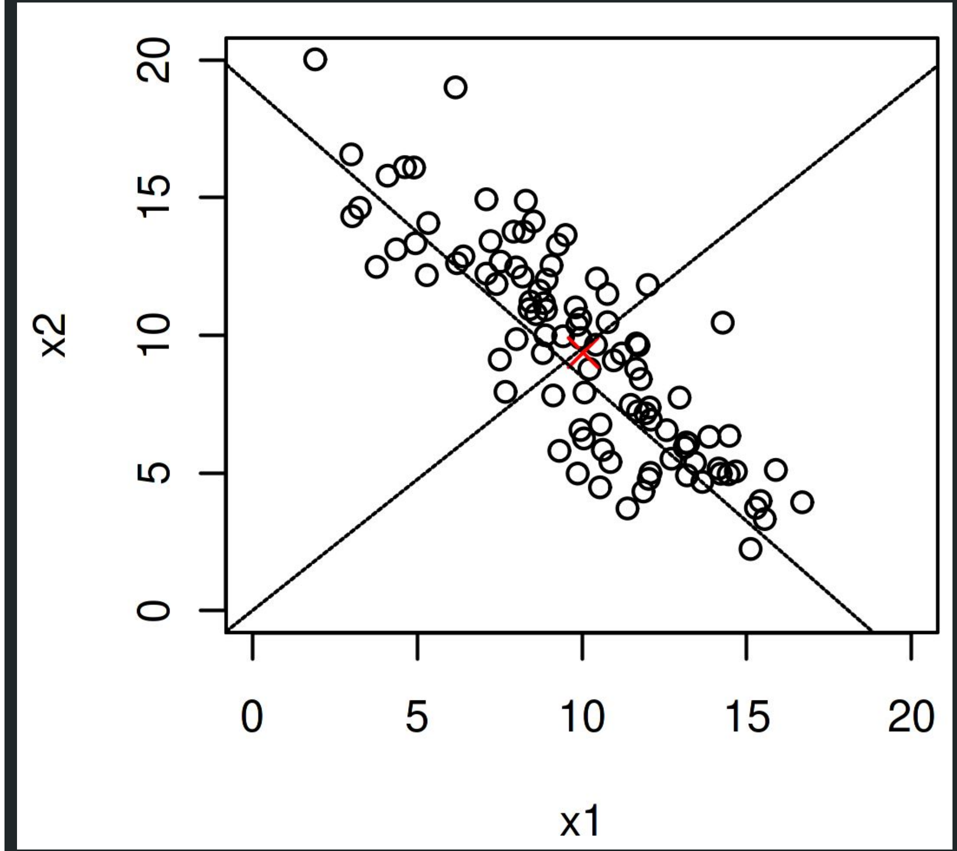
## Lab 2: Linguistics Data

- Good work!
- General tip when plotting: try not to use too many colors in discrete color scheme
    - Around 8 max, depending on the color scheme
- Even though unanswered questions are given to you as 0s, these should be treated as NAs
- Be careful when aggregating data
- Be careful with choice of centering / scaling

# PCA on uncentered data

This plot shows some 2D data (x1 and x2) and the principal component vectors for the data.

*Which line is the first PC and which is the second?*

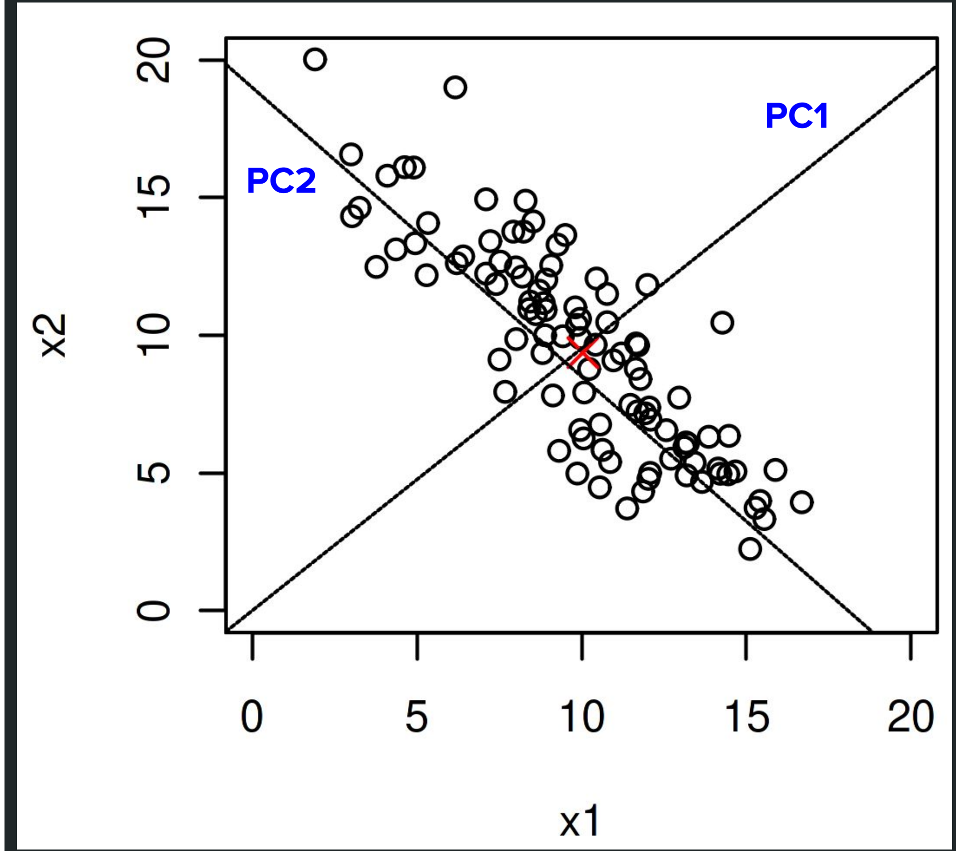


# PCA on uncentered data

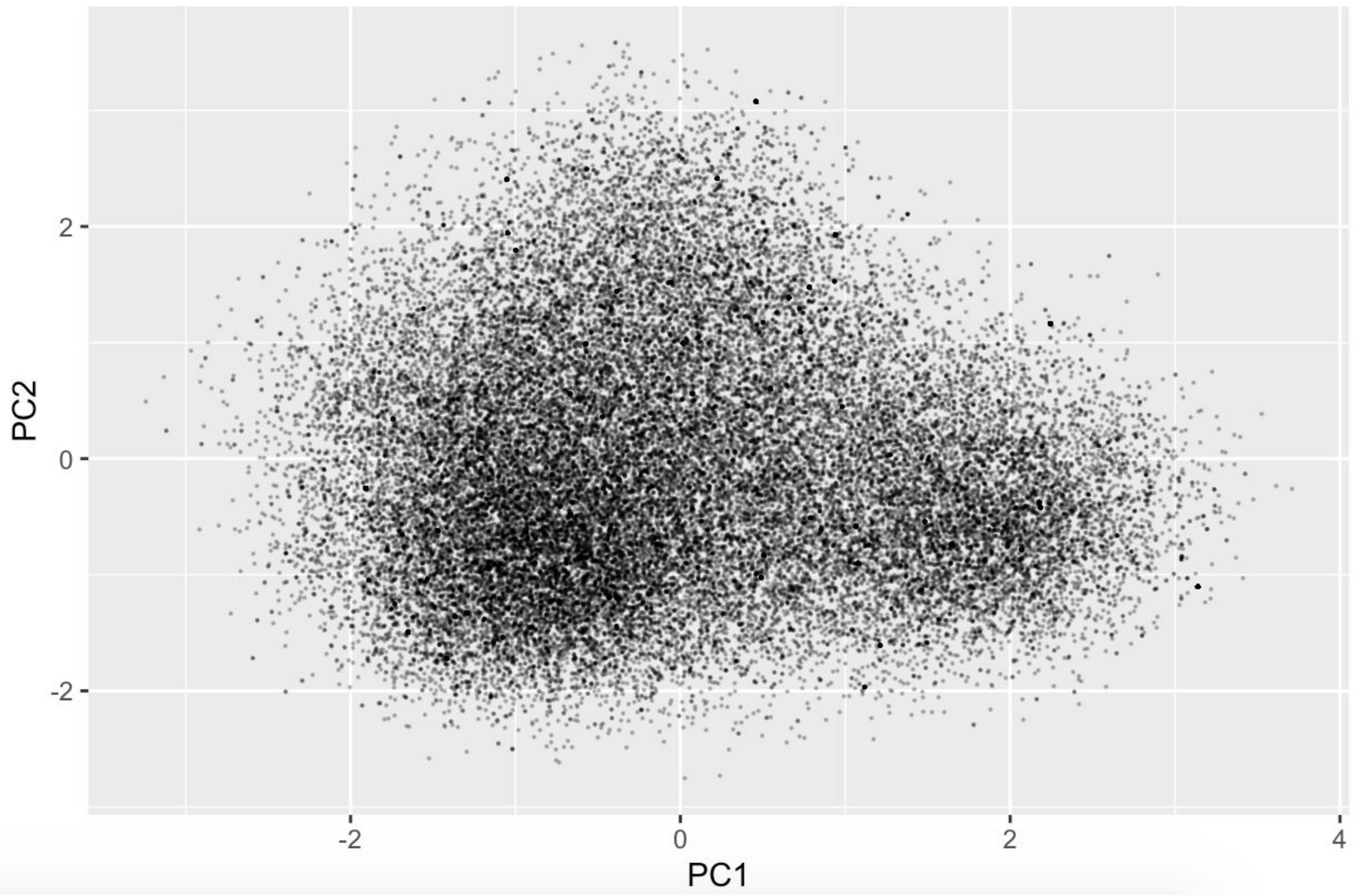
This plot shows some 2D data (x1 and x2) and the principal component vectors for the data.

*Which line is the first PC and which is the second?*

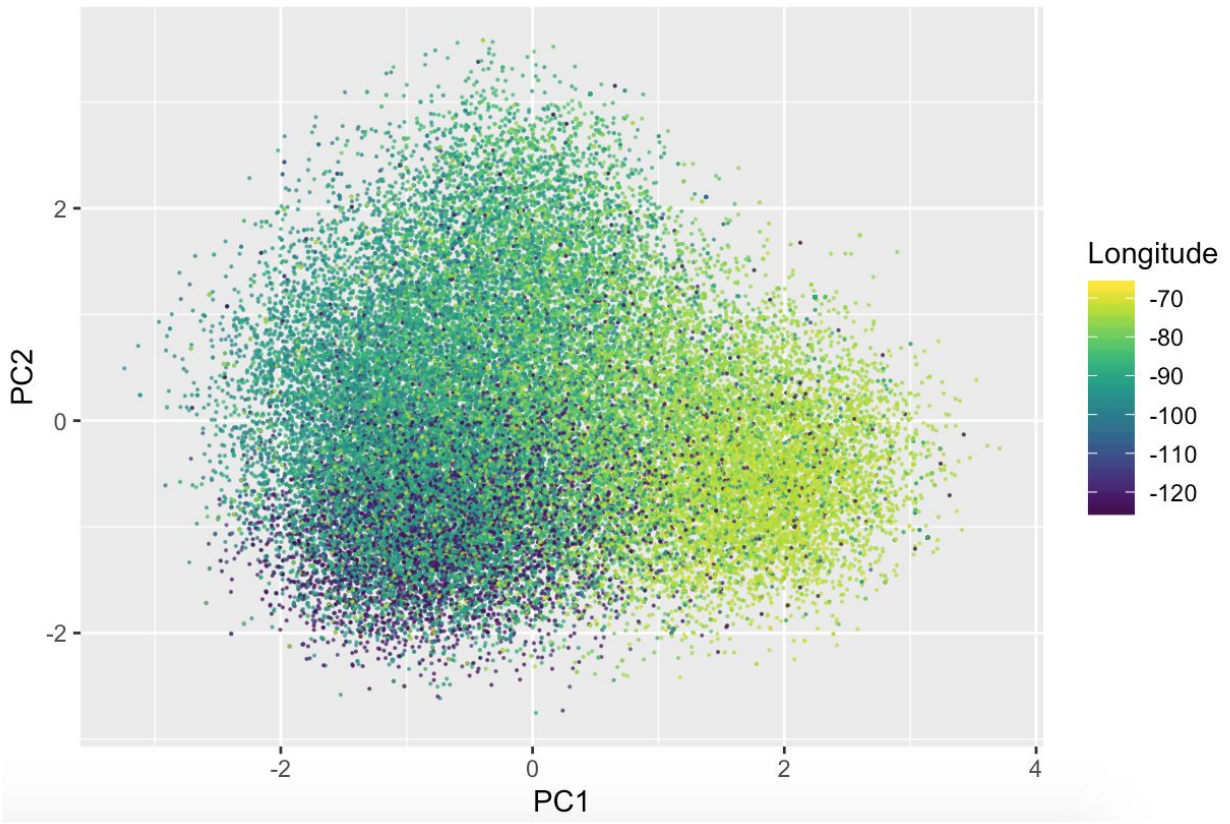
Does this surprise you?



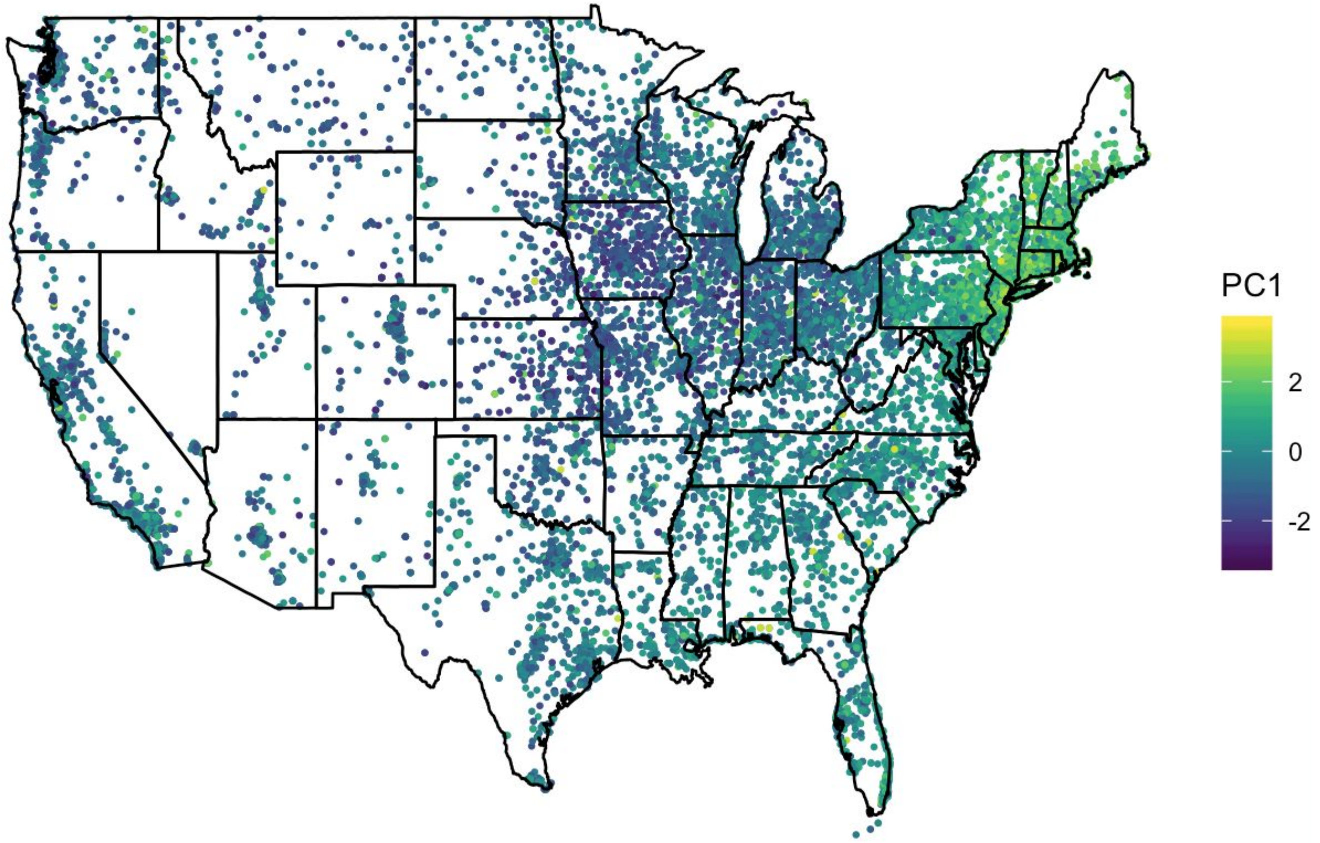
## PCA on lingData



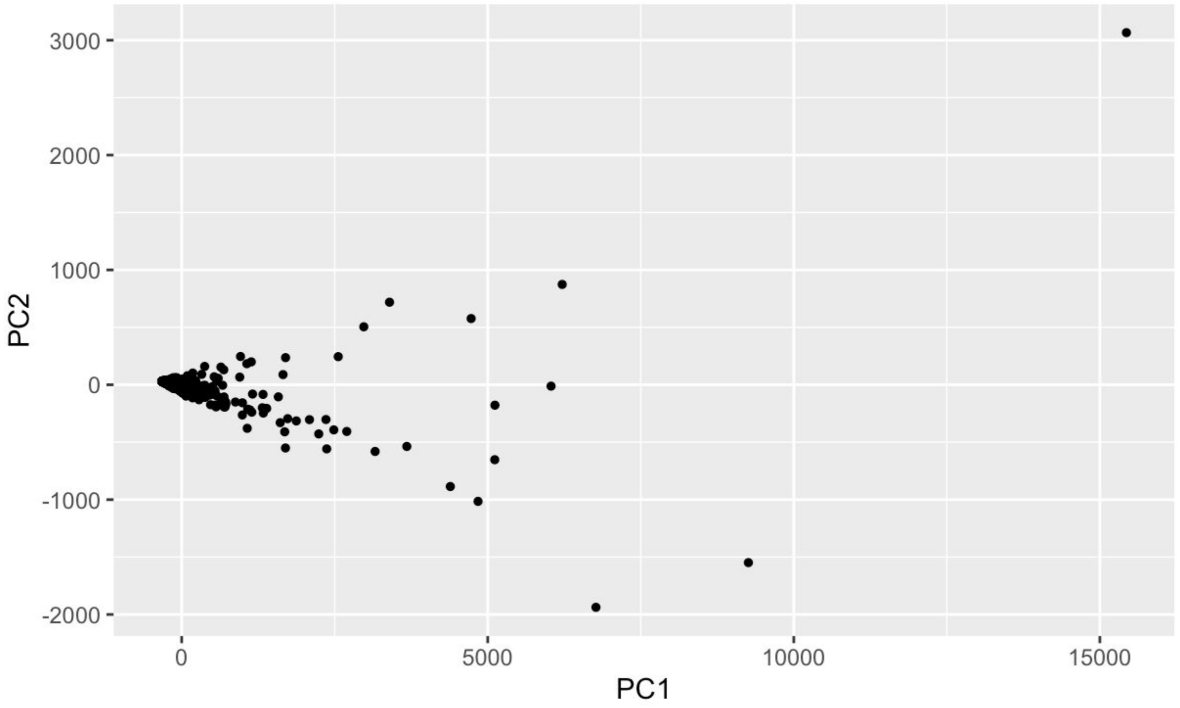
# PCA on lingData



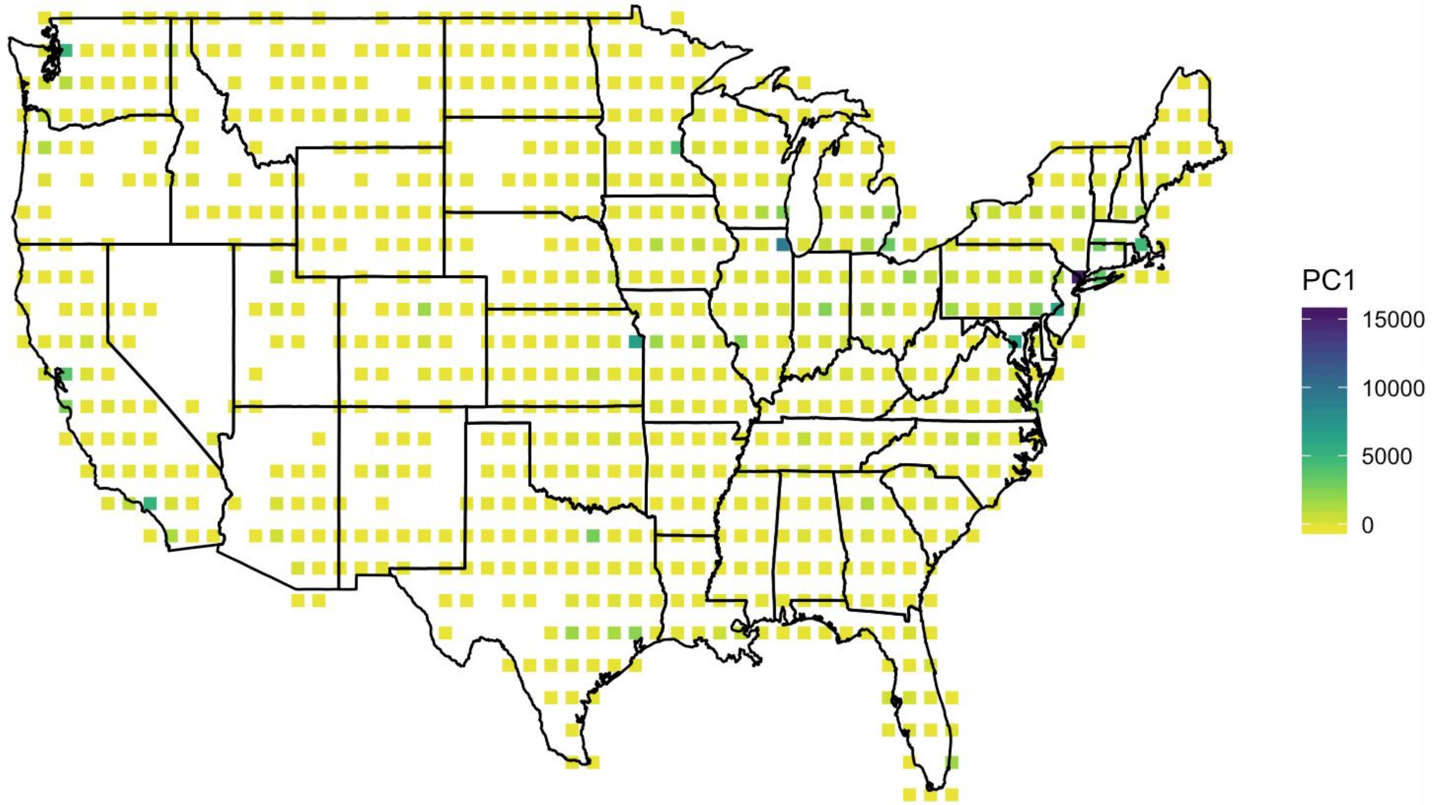
## PCA on lingData



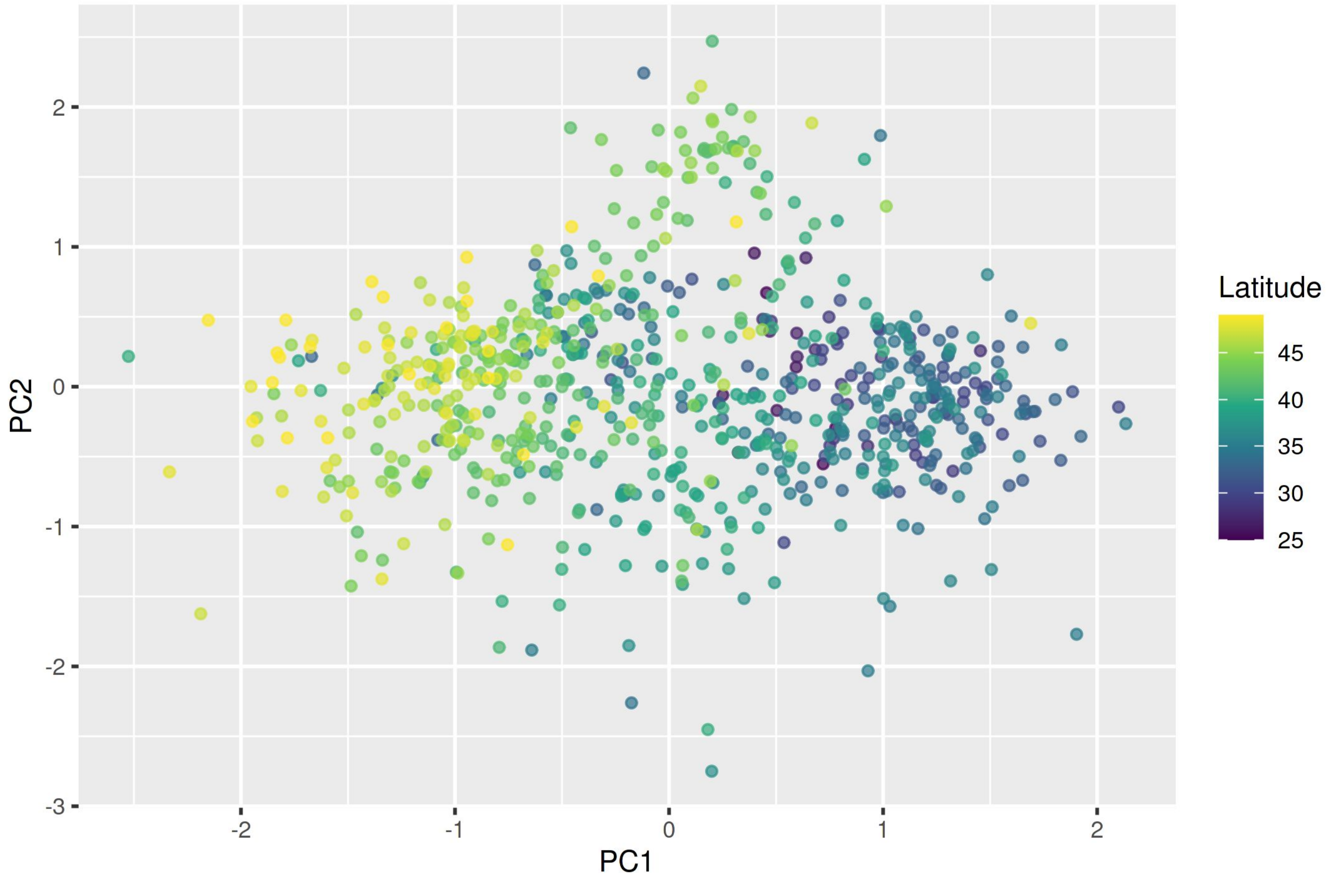
## PCA on lingLocation



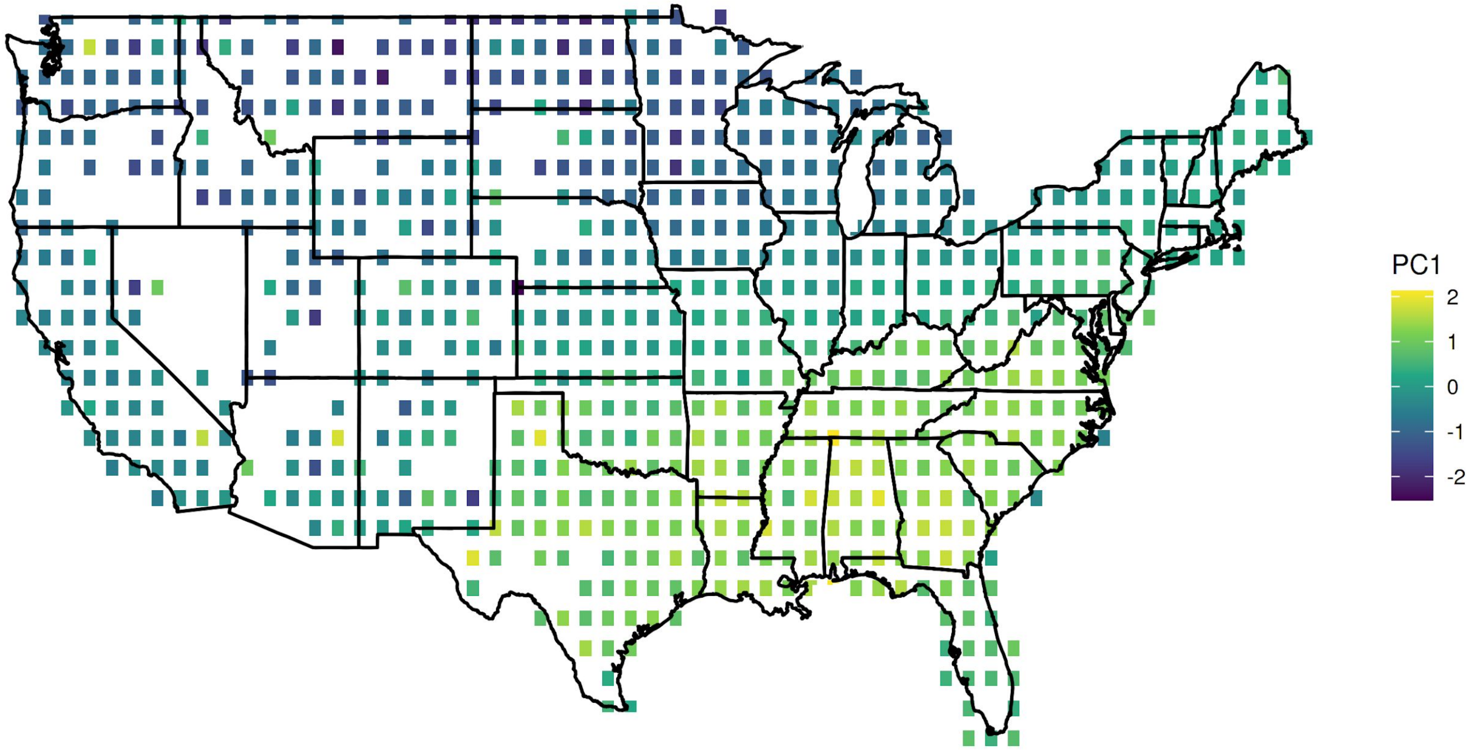
# PCA on lingLocation



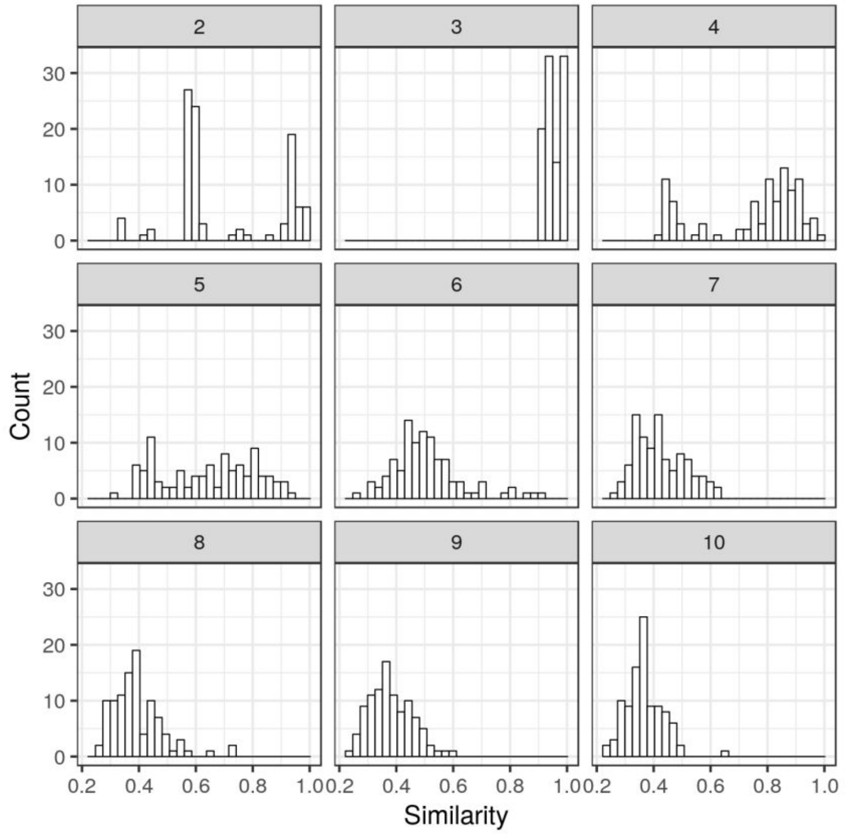
# PCA on lingLocation (after normalizing by # of responses)

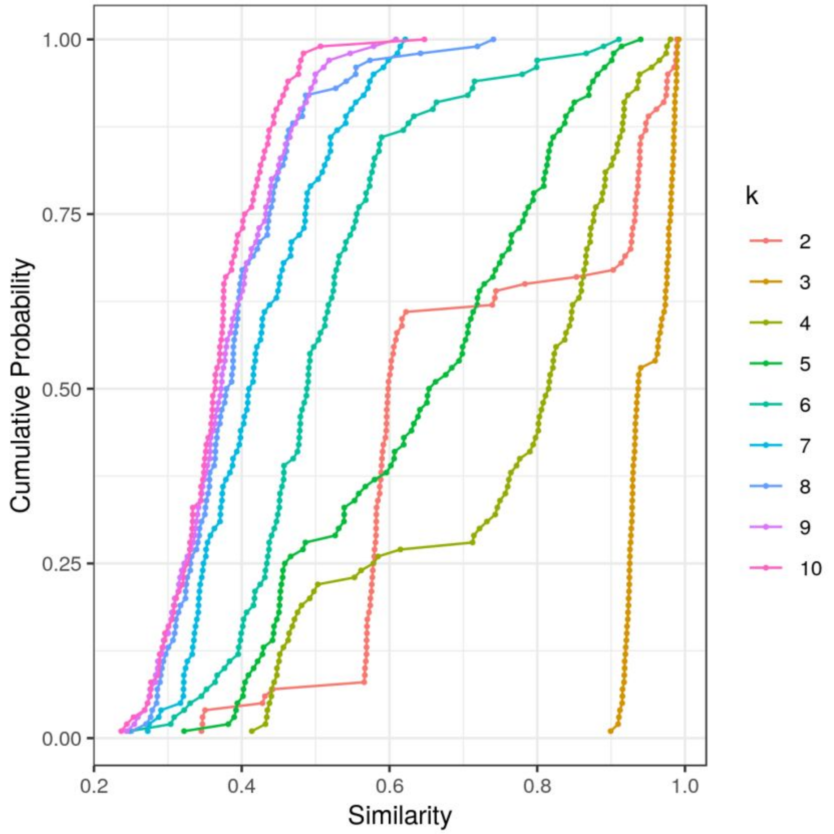


# PCA on lingLocation (after normalizing by # of responses)



# Lab 3: Stability of K-means





## Regularization Part II: Lasso

## Lasso motivation

**Feature selection**

- Want to find best subset of size $s$ of variables for prediction (typically $s$ << $p$)
  - Best for our purposes = smallest MSE
- Ridge doesn't perform *automatic* feature selection
  - Can use to order the importance of features via magnitude of their coefficients
  - Typically there are no exact 0's among the coefficients estimates
- Methods that provide more direct feature selection:
  - Best subsets ($L^0$) regression
  - Lasso

# Best subsets regression

**Goal**: Find the best (smallest MSE) subset of size $s$ of the features.

$$\tilde{\beta} = \underset{\beta}{\operatorname{argmin}} \left\{ \sum_{i=1}^{N} \left( y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \right)^2 + \lambda \underbrace{\sum_{j=1}^{p} |\beta_j|^0}_{\text{Number of non-zero coefficients}} \right\}$$

**Problem**:

- Computationally infeasible for moderate-sized problems[1]
- Have to do an exhaustive search over $\binom{p}{s}$ regressions

1. E.g. the R package `best-subset` can find solutions for p = 100 in a reasonable amount of time (but much slower than e.g. lasso)

## Lasso

Uses an $\boldsymbol{L^1}$ penalty (rather than the $L^2$ penalty that ridge uses).

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \Big( y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \Big)^2 \quad \text{subject to} \quad \sum_{j=1}^{p} |\beta_j| \le t.$$

Equivalently: $$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\operatorname{argmin}} \left\{ \frac{1}{2} \sum_{i=1}^{N} (y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right\}$$

- Lasso is the "best convex relaxation" of the best subsets problem
- The $L^1$ penalty results in **sparse** solutions (many coefficients zero)
- More interpretable than ridge, but if the true data-generating process is not sparse then can result in worse prediction accuracy

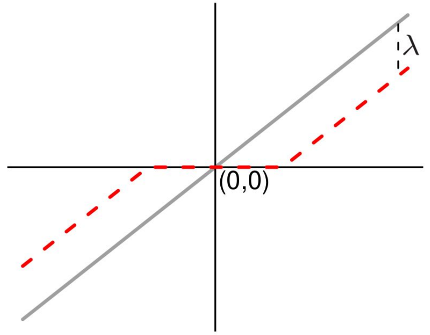
# Lasso with orthonormal design matrix

If the columns of the design matrix predictors X are orthonormal, then the lasso estimate is related to the OLS estimate by

$$\hat{\beta}_j^{\text{lasso}} = \text{sign}(\hat{\beta}_j)(|\hat{\beta}_j| - \lambda)_+$$

OLS estimate

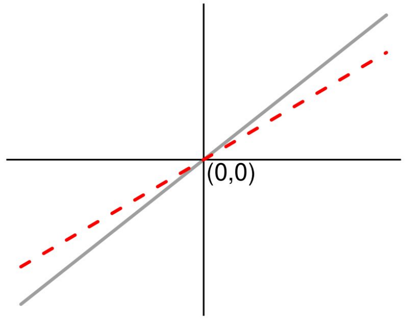
Lasso



Compare that to what happens in the case of ridge:

$$\hat{\beta}_j^{\text{ridge}} = \hat{\beta}_j/(1+\lambda)$$

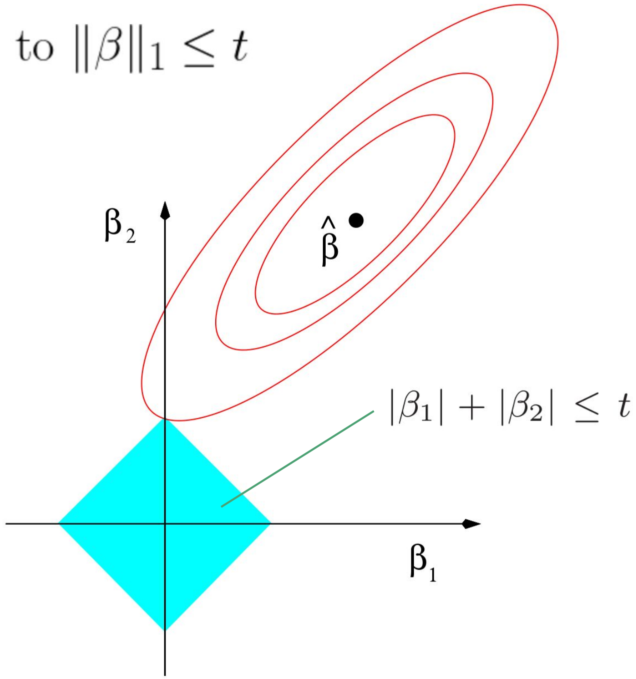
Ridge



# Lasso vs. ridge, non-orthogonal case

$$\hat{\beta}^{\text{lasso}} = \arg\min_{\beta} \|y - X\beta\|_2^2$$

subject to $\|\beta\|_1 \leq t$



$$\hat{\beta}^{\text{ridge}} = \arg\min_{\beta} \|y - X\beta\|_2^2$$

subject to $\|\beta\|_2^2 \leq t$

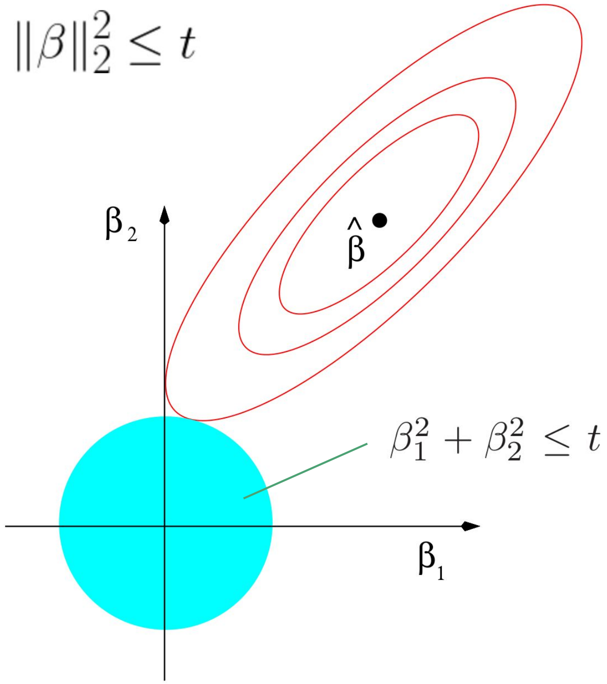
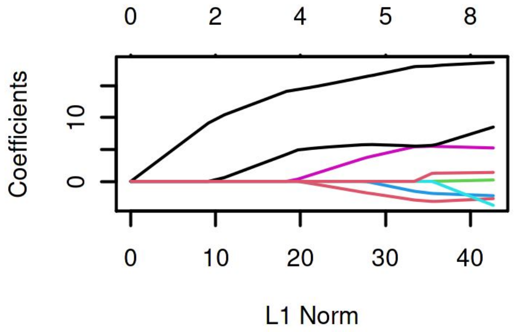


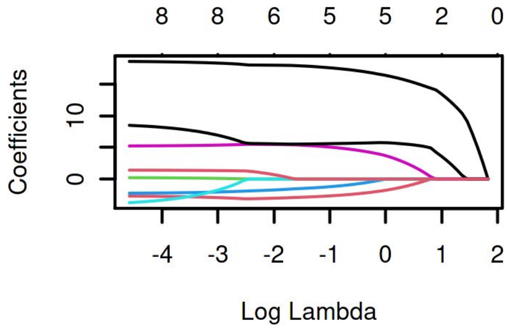
Image source: *The Elements of Statistical Learning*, p. 71

# Lasso shrinkage profiles

$$\hat{\beta}^{\text{lasso}} = \underset{\beta}{\text{argmin}} \sum_{i=1}^{N}\Big(y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j\Big)^2 \text{ subject to } \sum_{j=1}^{p}|\beta_j| \le t.$$

- If $t$ is larger than $t_0 = \sum_1^p |\hat{\beta}_j|$ then the lasso solution is the same as the OLS solution.
- If $t = t_0/2$ then lasso shrinks the OLS coefficients by about 50% on average
- By default, the R package `glmnet` plots the coefficients vs. their $L^1$ norm, but you can have it plot $\lambda$ on the $x$-axis instead.





# non-zero coefficients

## Lasso in practice

- Great for feature selection and interpretability
- Tends to select one representative from a group of correlated features
  - Good if goal is a parsimonious model
  - Bad for data-driven discoveries
- Like ridge, don't forget to center and scale data
  - `glmnet()` does this by default, but returns results on original scale
- For large $\lambda$, the coefficients are heavily biased
  - Eventually the variance goes to zero (all 0 coefficients)
- **Warnings**:
  - Non-zero lasso coefficients ≠ statistically significant
  - Don't fit OLS after doing lasso and interpret the p-values the same way; OLS inference no longer valid after model model selection

# Other alternatives / modifications to lasso

- **Weighted lasso**: use the weighted penalty $\lambda \sum_{j=1}^{p} w_j |\beta_j|$
    - If true $\beta_j$ is large (small), want to penalize less (more) so $w_j$ should be small (large)
- **Adaptive lasso** (Zou, 2006): $w_j = \left( \frac{1}{\left| \hat{\boldsymbol{\beta}}_j^{OLS/ridge} \right|} \right)^{\gamma} \quad (\gamma \in \{0.5, 1, 2\})$
- Non-convex penalties:
    - **SCAD** (Fan, 2001)
    - **MCP** (Zhang, 2009)
- **Randomized lasso** (Meinshausen, 2010): combines stability via bootstrapping and randomized weights to handle correlated variables

## Elastic net

$$\hat{\beta}^{\text{enet}} = \underset{\beta}{\operatorname{argmin}} \sum_{i=1}^{N} \Big( y_i - \beta_0 - \sum_{j=1}^{p} x_{ij}\beta_j \Big)^2 + \lambda \sum_{j=1}^{p} (\alpha|\beta_j| + (1-\alpha)\beta_j^2)$$

- Compromise between ridge and lasso depending on choice of $\alpha \in [0, 1]$
  - Lasso: $\alpha = 1$
  - Ridge: $\alpha = 0$
- Can handle correlated features better than lasso, but enforces more sparsity than ridge
- Difficult to tune both $\alpha$ and $\lambda$ in practice

## Summary: OLS + Regularized Regression

| | OLS | Ridge | Lasso | Elastic Net |
|---|---|---|---|---|
| **Advantages** | • Simple<br>• Can do inference (with all the caveats) | • Good with correlated features<br>• MSE Existence Theorem (good for prediction) | • Feature selection and sparsity<br>• Interpretability | • Compromise between ridge and Lasso |
| **Disadvantages** | • Major problems when p>n | • Dense model is not interpretable<br>• No automatic feature selection | • Tends to choose 1 feature from correlated group<br>• Can give worse prediction results if truth is not sparse | • Difficult to tune two parameters in practice |

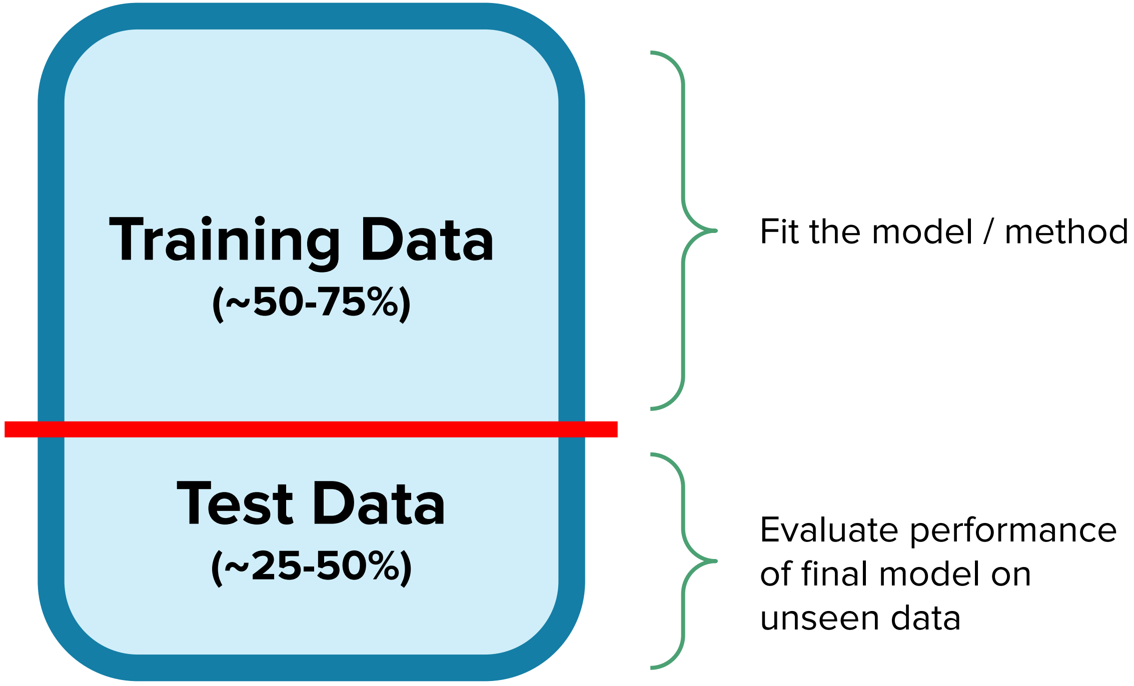
## Statistical learning

**Goal**: Learn insights about the current data that are *generalizable* to future data

- Typically, one of the following tasks in mind:
    - **Prediction**: outcome of interest
    - **Data-driven discoveries**: pattern recognition, feature selection, etc.
- Two main branches of statistical learning:
    - **Unsupervised**: no outcomes / response data
        - Clustering, dimension reduction, pattern recognition
    - **Supervised**: have the outcome / response
        - **Classification** - categorical response
        - **Regression** - continuous response

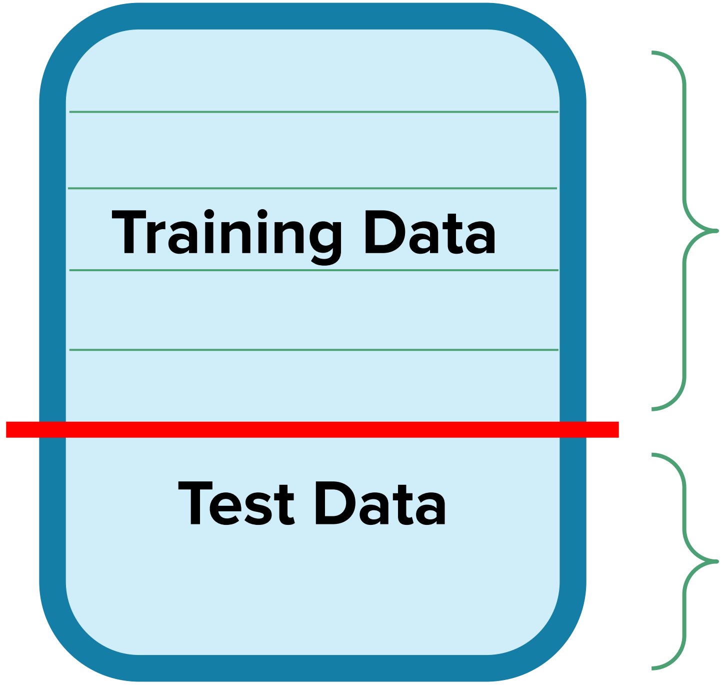
## Data splitting: a way to assess generalizability

Simplest case:



## Data splitting: a way to assess generalizability

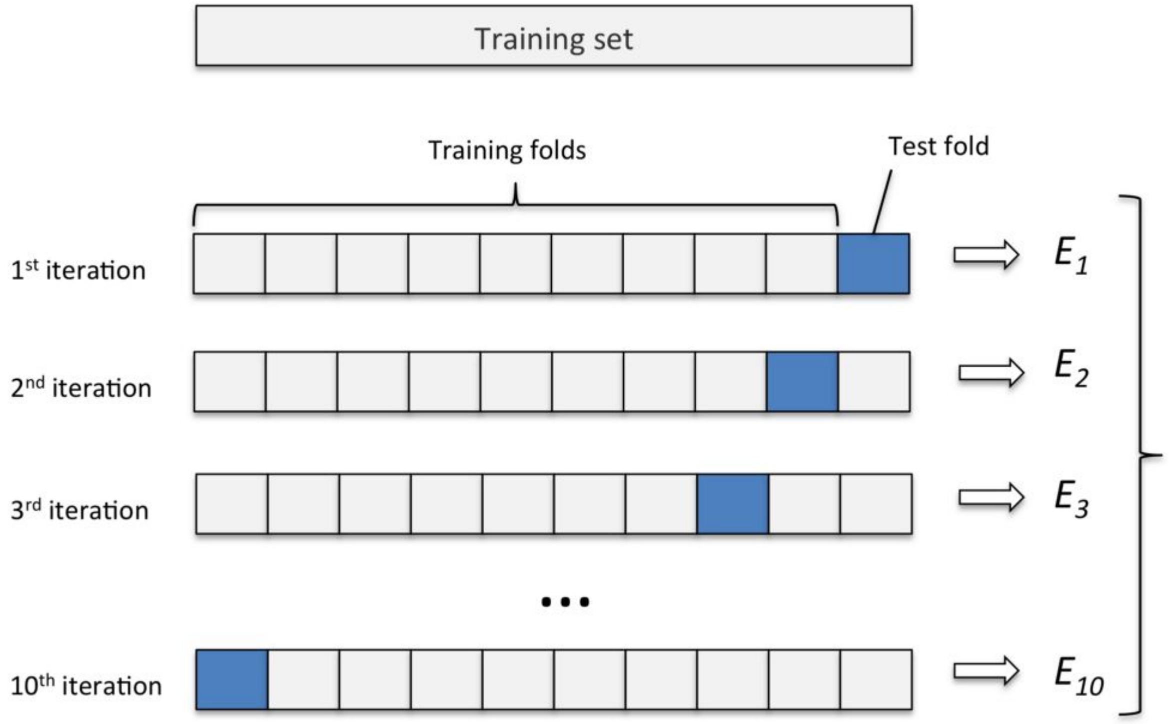
With cross validation:



Do 5-fold or 10-fold CV to choose hyperparameters and fit tuned model

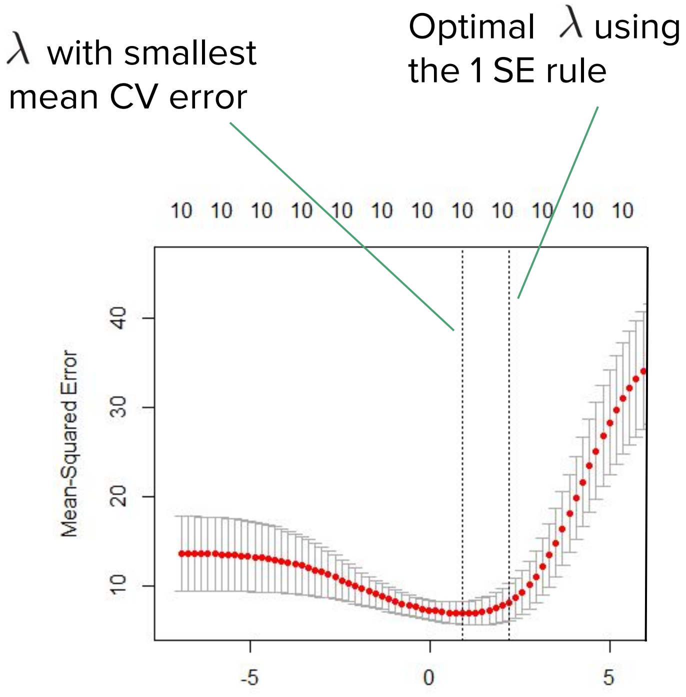
Evaluate performance of final model on unseen data

# K-fold cross validation for choosing hyperparameters



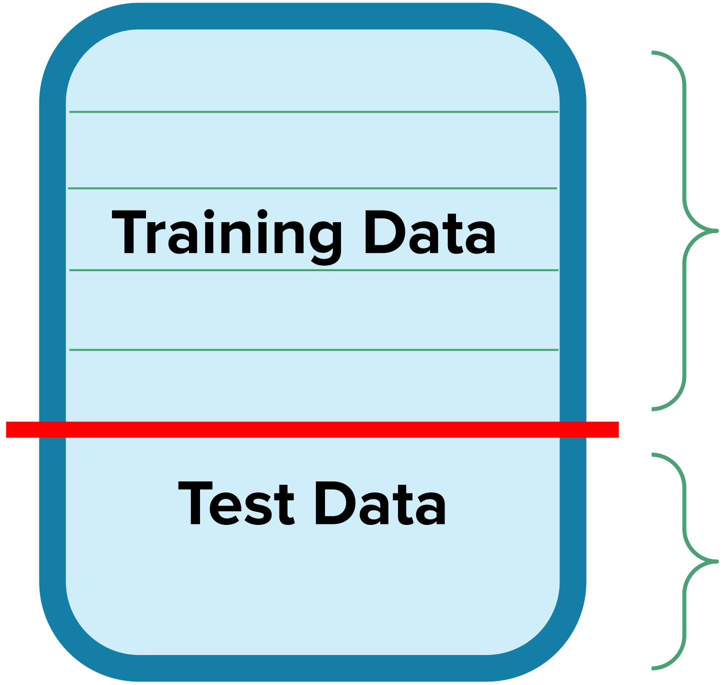
$\lambda$ with smallest mean CV error

Optimal $\lambda$ using the 1 SE rule



# Data splitting: a way to assess generalizability

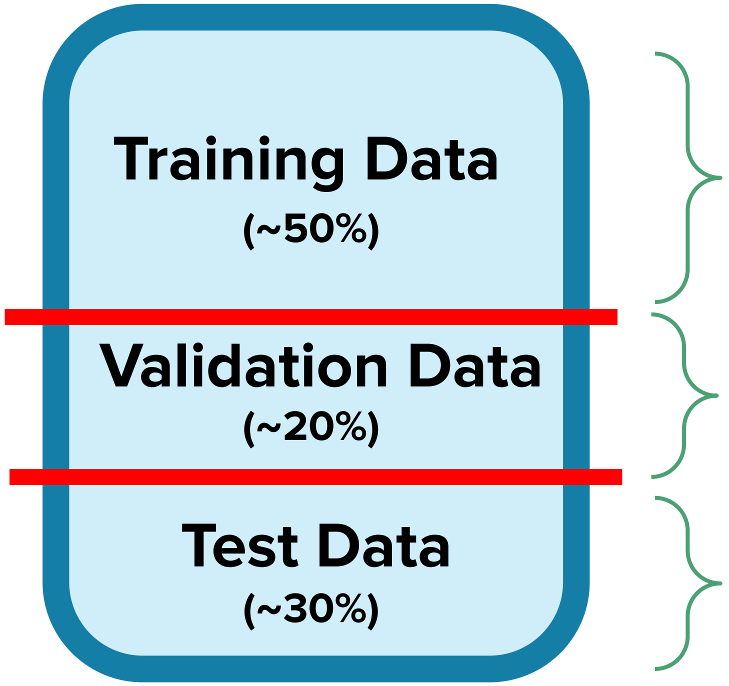
With cross validation:



Do 5-fold or 10-fold CV to choose hyperparameters and fit tuned model

Evaluate performance of final model on unseen data

## Data splitting + tuning hyperparameters + multiple methods



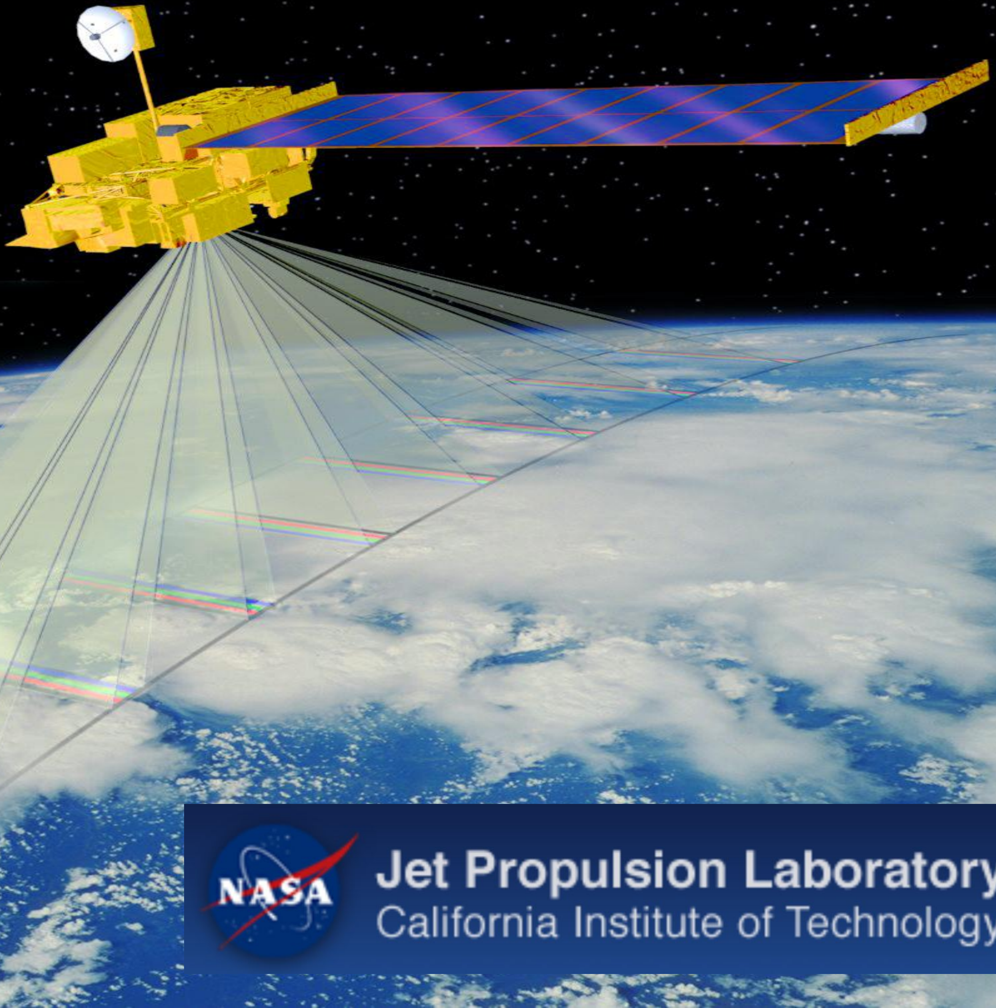
E.g., use CV on this split to pick best $\lambda$ for both lasso and ridge

Evaluate trained lasso and ridge models and pick the model with lower error

Evaluate performance of final model on unseen data

- Can repeat this data splitting B times to get a variance estimate of the test error
- This gives you an unbiased estimate of the prediction error for the *statistical learning process*, NOT a specific model

# Lab 4: Cloud Data

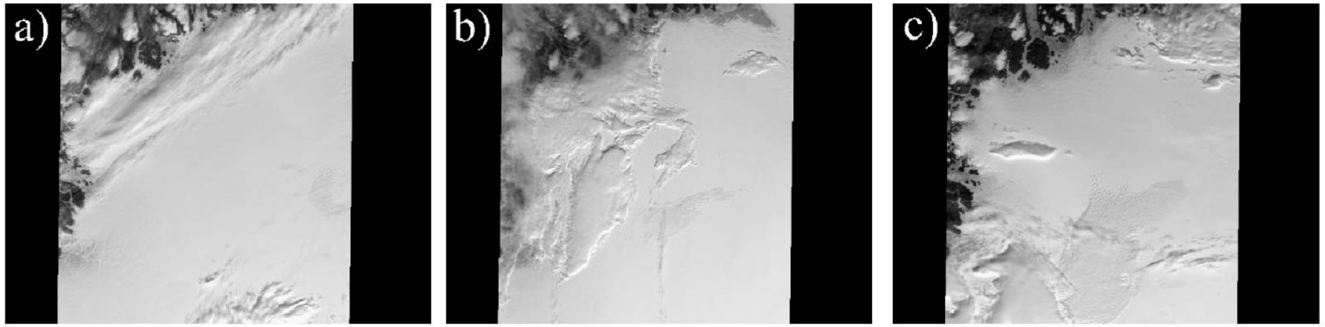


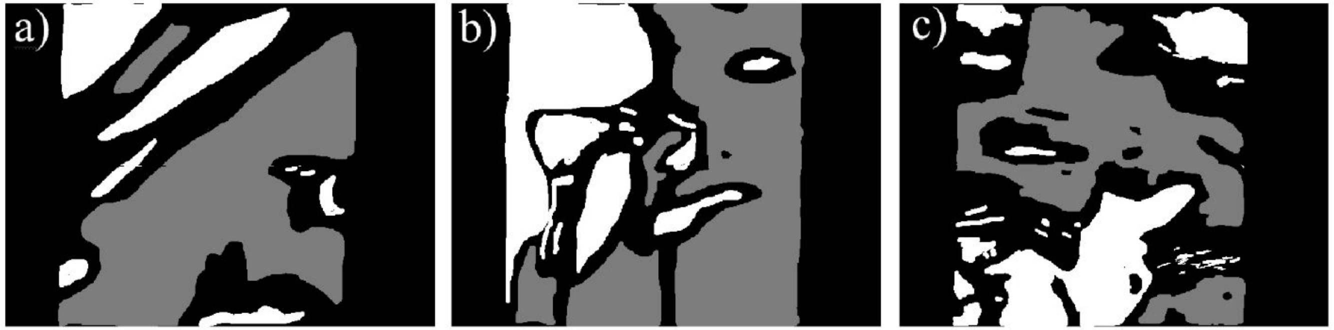
MISR

Multi-angle Imaging SpectroRadiometer

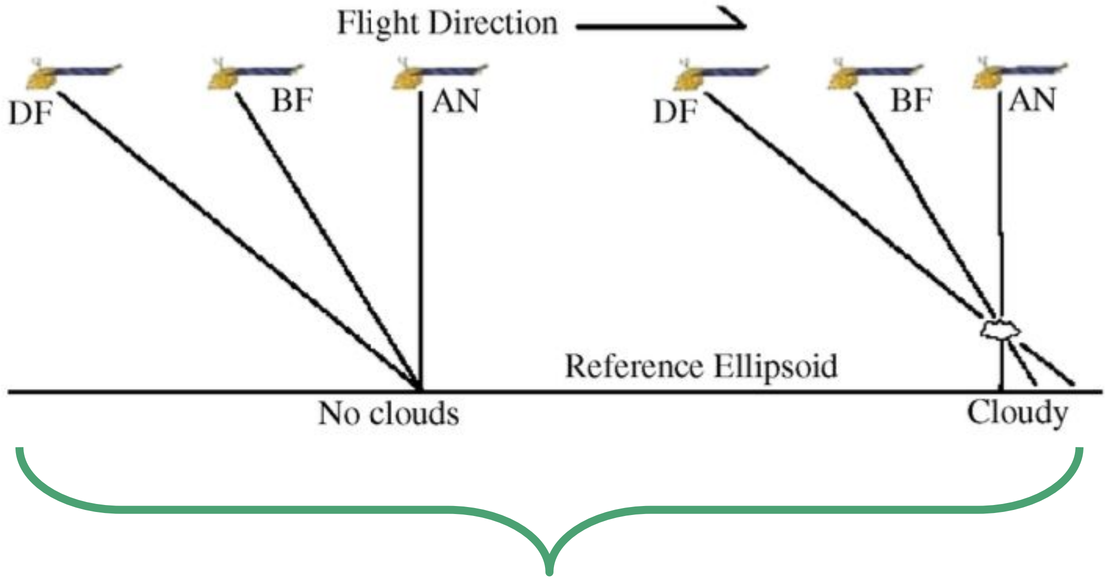
NASA Jet Propulsion Laboratory
California Institute of Technology

# Lab 4: Remote sensing / cloud data





## Lab 4: Remote sensing / cloud data



Feature engineering:
CORR, SD, NDAI

## Lab 4: Things to think about carefully

- Which methods/models?
  - Are they well-suited for this data? Why or why not?
  - What are the advantages/disadvantages and assumptions of the method(s) that you chose?
  - This can help you better identify the limitations of your prediction algorithm
- Data splitting scheme? Very important for generalizability
- Post-hoc EDA? Can provide insights into how to improve your prediction

## Let’s implement this pipeline in R

Using the `ozone` data:

1. Pick two (or more) regression methods and an error metric
   - E.g. ridge, lasso, elastic net, SCAD, MCP, etc.
2. Split the data into training / validation / test sets
3. On the training set, use CV to tune hyperparameters for each method under consideration
4. Evaluate the tuned methods on the validation set
5. Report the test error of your “best” method
6. Time permitting, repeat steps 2-5 B times times to get a variance estimate
7. Time permitting, perform EDA to diagnose the errors - are there any observations that are particularly difficult to predict? Any common mistakes?