LECTURE 03: LINEAR REGRESSION PT. 1

September 18, 2017

SDS 293: Machine Learning

https://www.science.smith.edu/~jcrouser/SDS293/

Residual standard error

- **Idea**: estimate standard deviation of ϵ using RSS to get ***residual standard error***:

$$RSE = \sqrt{\frac{RSS}{(n-2)}}$$

- Now we can finally estimate SE, which can be used to compute ***confidence intervals***
- In linear regression, the 95% confidence intervals are:

$$\hat{\beta}_0 \pm 2\times SE\left(\hat{\beta}_0\right) \text{ and } \hat{\beta}_1 \pm 2\times SE\left(\hat{\beta}_1\right)$$

LECTURE 10:

LINEAR MODEL SELECTION PT. 1

October 16, 2017

SDS 293: Machine Learning

Outline

- Model selection: alternatives to least-squares
- Subset selection
  - Best subset
  - Stepwise selection (forward and backward)
  - Estimating error
- Shrinkage methods
  - Ridge regression and the Lasso
  - Dimension reduction
- Labs for each part

Back to the safety of linear models…

$$\mathtt{Y} \approx \beta_0 + \beta_1 \mathtt{X}_1 + \ldots + \beta_p \mathtt{X}_p$$

Bias vs. variance

Discussion

How could we

reduce the variance?

Subset selection

- **Big idea:** if having too many predictors is the problem maybe we can get rid of some
- **Problem:** how do we choose?

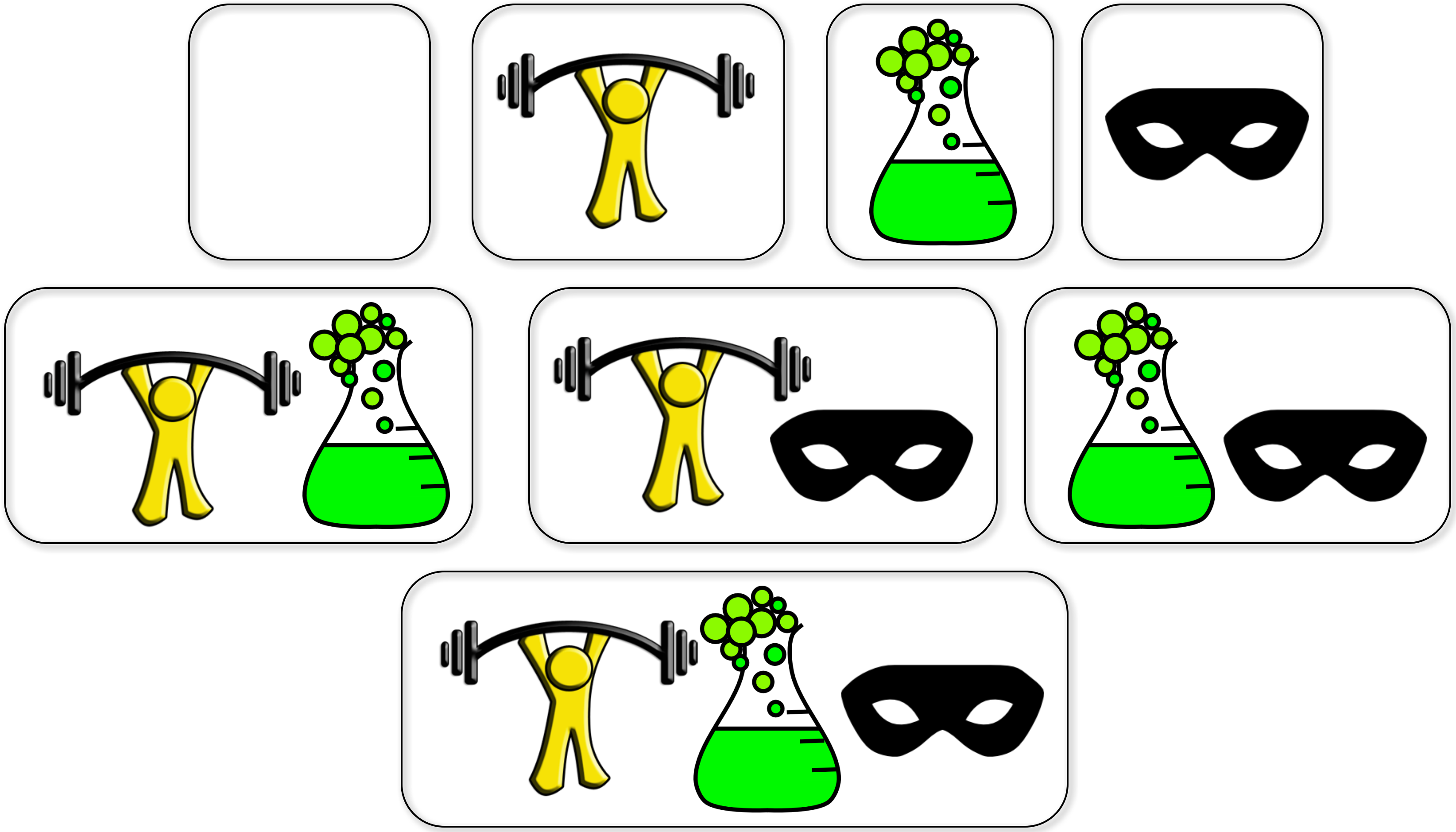
Flashback: superhero example

7"
6"
5"
4"
3"
2"
1"
OLLIE! EYES FRONT!!!
DO YOU BLEED?
STOP BEING CREEPY!
ROBIN 4' 6"
BATGIRL 5' 3"
SUPERGIRL 5' 5"
HAWKGIRL 5' 6"
CATWOMAN 5' 7"
BLACK CANARY 5' 7"
NIGHTWING 5' 10"
GREEN ARROW 5' 11"
POWER GIRL 5' 11"
THE FLASH 6' 0"
WONDER WOMAN 6' 0"
THE ATOM 6' 0"
GREEN LANTERN 6' 0"
AQUAMAN 6' 1"
SHAZAM 6' 2"
BATMAN 6' 2"
SUPERMAN 6' 3"
STARFIRE 6' 4"
CYBORG 6' 6"
MARTIAN MANHUNTER 6' 7"
SWAMP THING VARIES

$$height = \beta_1\left(\quad\right) + \beta_2\left(\quad\right) + \beta_3\left(\quad\right)$$

Image credit: Ming Malaykham

Best subset selection: try them all!



Finding the "best" subset

Start with the null model M_0 (containing no predictors)

1. For $k = 1,2, \ldots, p$:
   a. Fit all (p choose k) models that contain exactly p predictors.
   b. Keep only the one that has the smallest RSS (or equivalently the largest R^2). Call it M_k.
2. Select a single "best" model from among $M_0 \ldots M_p$ using cross-validated prediction error or something similar.

Discussion

Question 1: why not just use the one with the lowest RSS?

Answer: because you'll always wind up choosing the model with the highest number of predictors (why?)

Discussion

Question 2: why not just calculate the cross-validated prediction error on all of them?

Answer: so… many... models...

A sense of scale…

- We do a lot of work in groups in this class
- How many different possible groupings are there?
- Let’s break it down:

47 individual people

1,081 different groups of two

16,215 different groups of three…

Model overload

- Number of possible models on a set of p predictors:

$$\sum_{k=1}^{p} \binom{p}{k} = 2^p$$

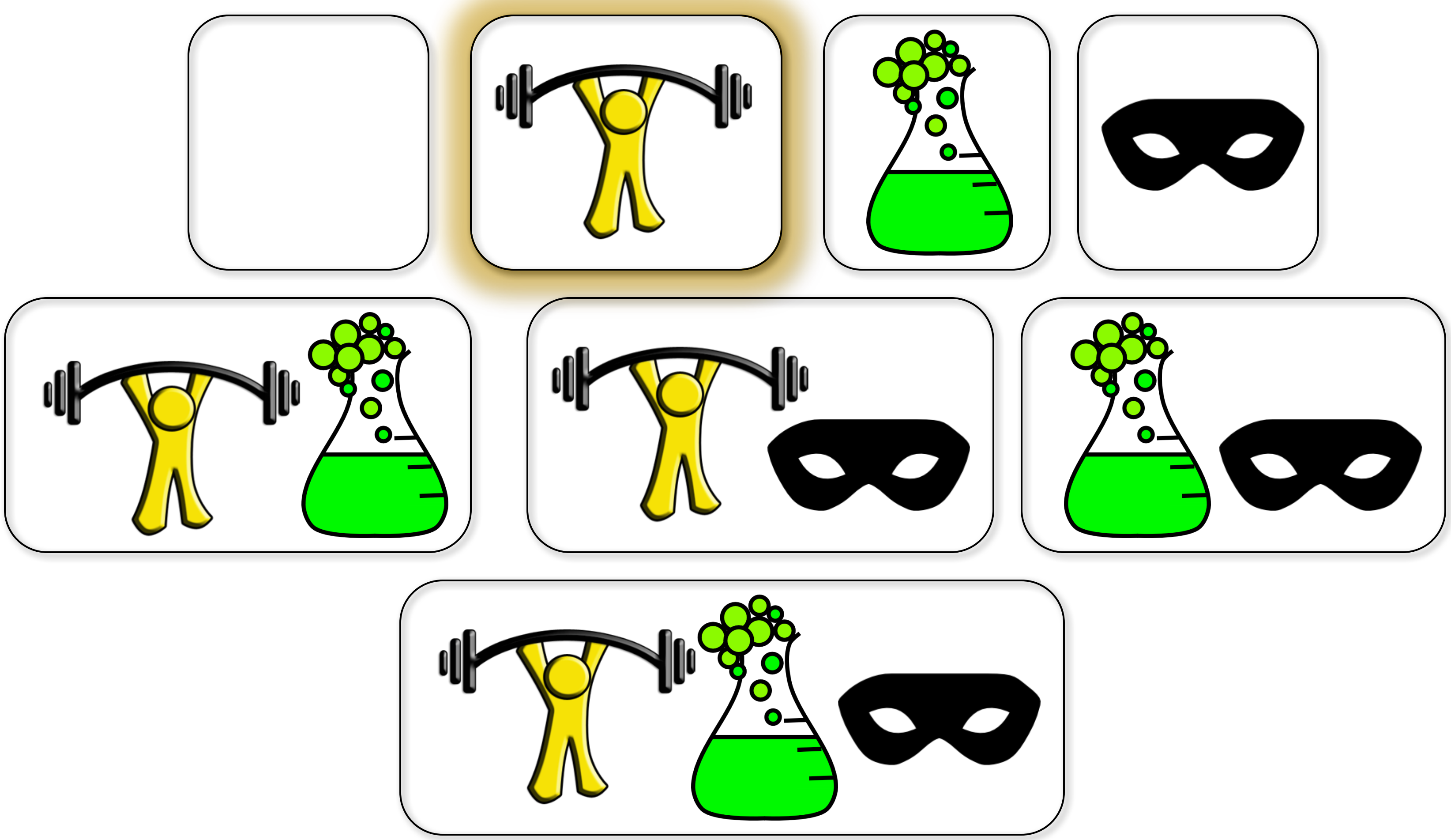
- On 10 predictors: **1,024** models
- On 20 predictors: **1,048,576** models

A bigger problem

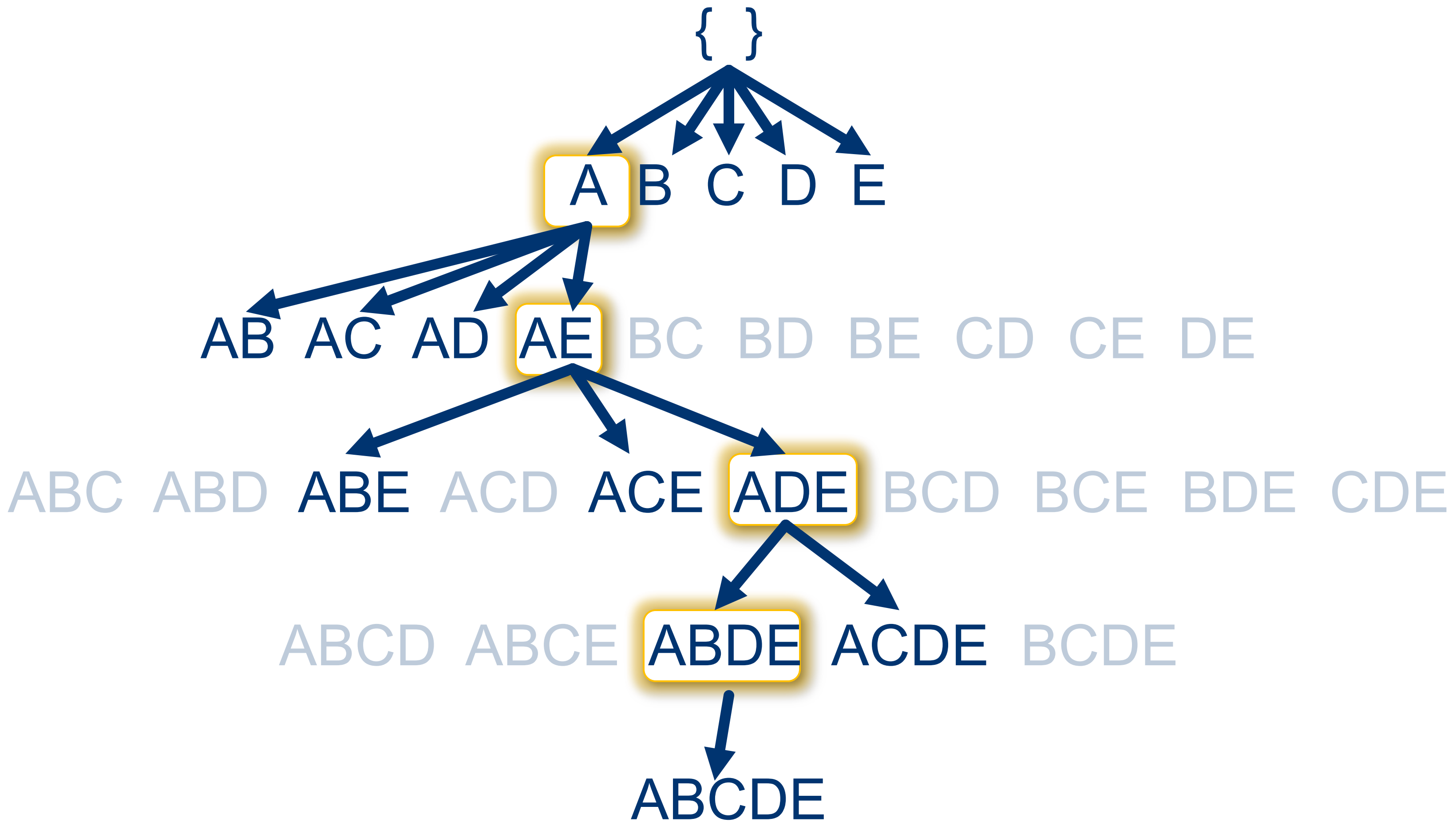
Question: what happens to our estimated coefficients as we fit more and more models?

Answer: the larger the search space, the larger the variance. We're overfitting!

What if we could eliminate some?



A slightly larger example ($p = 5$)



Best subset selection

Start with the null model M_0 (containing no predictors)

1. For $k = 1, 2, \dots, p$:
   a. Fit all $(p\ choose\ k)$ models that contain exactly p predictors.
   b. Keep only the one that has the smallest RSS (or equivalently the largest R^2). Call it M_k.
2. Select a single “best” model from among $M_0 \dots M_p$ using cross-validated prediction error or something similar.

Forward selection

Start with the null model M_0 (containing no predictors)

1. For $k = 1, 2, \ldots, p$:
   a. Fit all $(p - k)$ models that augment M_{k-1} with exactly 1 predictor.
   b. Keep only the one that has the smallest RSS (or equivalently the largest R^2). Call it M_k.
2. Select a single "best" model from among $M_0 \ldots M_p$ using cross-validated prediction error or something similar.

Stepwise selection: way fewer models

- Number of models we have to consider:

$$\sum_{k=1}^{p} \binom{p}{k} = 2^p \rightarrow \sum_{k=0}^{p-1} (p-k) = 1 + \frac{p(p+1)}{2}$$

- On 10 predictors: 1024 models → **51 models**
- On 20 predictors: over 1 million models → **211 models**

Forward selection

Question: what potential problems do you see?

Answer: there's a risk we might prune an important predictor too early. While this method usually does well in practice, it is not guaranteed to give the optimal solution.

Forward selection

Start with the null model M_0 (containing no predictors)

1. For $k = 1,2,\dots,p$:
   a. Fit all $(p-k)$ models that augment M_{k-1} with exactly 1 predictor.
   b. Keep only the one that has the smallest RSS (or equivalently the largest R^2). Call it M_k.
2. Select a single “best” model from among $M_0 \dots M_p$ using cross-validated prediction error or something similar.

Backward selection

Start with the full model M_p (containing all predictors)

1. For $k = p, (p-1), \ldots, 1$:
   a. Fit all k models that reduce M_{k+1} by exactly 1 predictor.
   b. Keep only the one that has the smallest RSS (or equivalently the largest R^2). Call it M_k.
2. Select a single "best" model from among $M_0 \ldots M_p$ using cross-validated prediction error or something similar.

Forward selection

Question: what potential problems do you see?

Answer: if we have more predictors than we have observations, this method won't work (why?)

Choosing the optimal model

- Flashback: measures of **training** error (RSS and R^2) aren't good predictors of **test** error (what we care about)
- Two options:
  1. We can **directly** estimate the test error, using either a validation set approach or cross-validation
  2. We can **indirectly** estimate test error by making an adjustment to the training error to account for the bias

Adjusted R^2

- **Intuition:** once all of the useful variables have been included in the model, adding additional junk variables will lead to only a small decrease in RSS

$$R^2 = 1 - \frac{RSS}{TSS} \to R^2_{Adj} = 1 - \frac{RSS/(n-d-1)}{TSS/(n-1)}$$

- Adjusted R^2 pays a penalty for unnecessary variables in the model by dividing RSS by $(n-d-1)$ in the numerator

AIC, BIC, and C_p

- Some other ways of penalizing RSS

Estimate of the variance of the error terms

$$C_p = \frac{1}{n}\left(RSS + 2d\boxed{\hat{\sigma}^2}\right)$$

$$AIC = \frac{1}{n\hat{\sigma}^2}\left(RSS + 2d\hat{\sigma}^2\right)$$

Proportional for least-squares models

$$BIC = \frac{1}{n}\left(RSS + \underbrace{\boxed{\log(n)d\hat{\sigma}^2}}_{}\right)$$

More severe penalty for large models

Adjust or validate?

Question: what are the benefits and drawbacks of each?

	Adjusted measures	**Validation**
Pros	Relatively **inexpensive** to compute	More **direct** estimate (makes fewer assumptions)
Cons	Makes more **assumptions** about the model – more opportunities to be wrong	More **expensive**: requires either cross validation or a test set

Lab: subset selection

- To do today's lab in R: **`leaps`**
- To do today's lab in python: **`itertools`**, **`time`**
- Instructions and code:

  [course website]/labs/lab8-r.html

  [course website]/labs/lab8-py.html
- Full version can be found beginning on p. 244 of ISLR

LECTURE 11:

LINEAR MODEL SELECTION PT. 2

October 18, 2017

SDS 293: Machine Learning

Flashback: subset selection

- **Big idea:** if having too many predictors is the problem maybe we can get rid of some

- Three methods:
  - **Best subset:** try all possible combinations of predictors
  - **Forward**: start with no predictors, greedily add one at a time
  - **Backward**: start with all predictors, greedily remove one at a time

"greedy" = Add/remove whichever predictor improves your model **right now**

Flashback: comparing methods

	Best Subset Selection	Forward Selection	Backward Selection
How many models get compared?	2^p	$1+\frac{p(p+1)}{2}$	$1+\frac{p(p+1)}{2}$
Benefits?	Provably optimal	Inexpensive	Inexpensive; doesn't ignore interaction
Drawbacks?	Exhaustive search is expensive	Not guaranteed to be optimal; ignores interaction	Not guaranteed to be optimal; breaks when $p>n$

Flashback: choosing the optimal model

- We know measures of training error (RSS and R^2) aren't good predictors of test error (what we actually care about)
- Two options:
  - We can **indirectly** estimate test error by making an adjustment to the training error to account for the bias:

$$R^2_{adj} \quad C_p \quad AIC \quad BIC$$

Pros: inexpensive to compute

Cons: makes additional assumptions about the model

  - We can **directly** estimate the test error, using either a validation set approach or a cross-validation approach

Discussion: potential problems?

Only training on a subset of the data means our model is **less accurate**

From the kitchen of: Grandma SDS

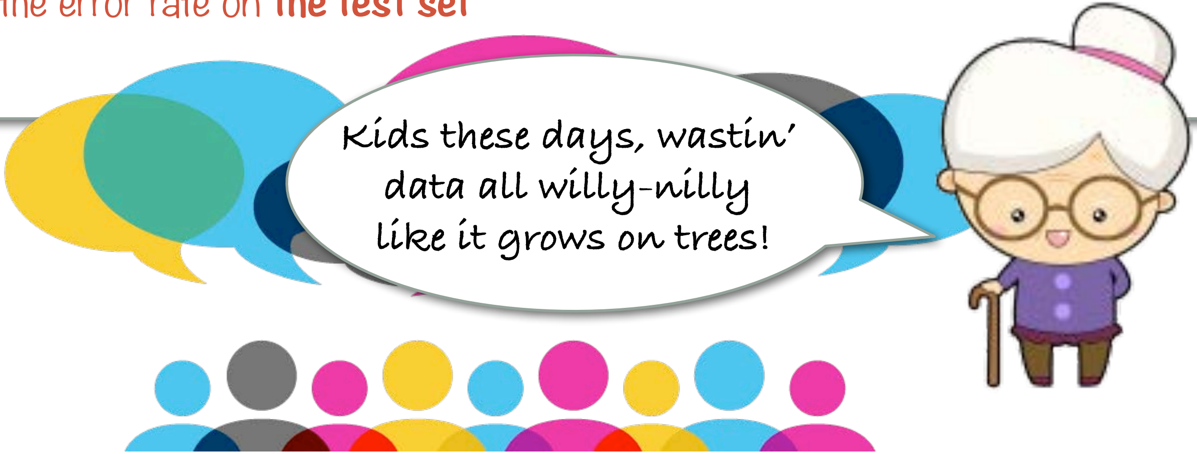
Recipe for: Best Subset Selection

First divide the data into training and test sets

Preheat the null model M_0 with no predictors.* on the **training set**

```
1.  For k = 1, 2, ..., p:
    a.  Fit all the models that contain exactly k predictors.
    b.  Keep only the model with the smallest training error. Call it M_k.

2.  ~~Estimate the error,~~ and select a single "best" model from among M_0 ... M_p
    ^ Calculate the error rate on the test set
```


Kids these days, wastin' data all willy-nilly like it grows on trees!

Cross-validation: how would this work?

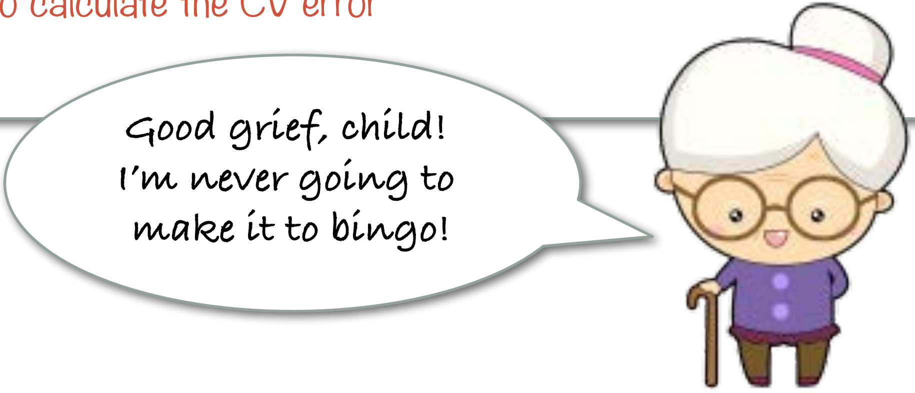
From the kitchen of: Grandma SDS

Recipe for: Best Subset Selection

Preheat the null model M_0 with no predictors.

1. For $k = 1, 2, \ldots, p$:
    a. Fit all the models that contain exactly k predictors.
    b. Keep only the model with the smallest training error. Call it M_k.

2. ~~Estimate the error,~~ and select a single "best" model from among $M_0 \ldots M_p$
^ Use k-fold cross-validation to calculate the CV error



Lab: subset selection using validation

- To do today's lab in R: <nothing new>
- To do today's lab in python: <nothing new>
- Instructions and code for part 1:

  http://www.science.smith.edu/~jcrouser/SDS293/labs/lab9.html
- Full version can be found beginning on p. 248 of ISLR
- For part 2:
  - Apply these techniques to a dataset of your choice
  - You're welcome (encouraged?) to work in teams!

LECTURE 12:
LINEAR MODEL SELECTION PT. 3

October 23, 2017

SDS 293: Machine Learning

Outline

- Model selection: alternatives to least-squares

✓Subset selection

  ✓Best subset

  ✓Stepwise selection (forward and backward)

  ✓Estimating error using cross-validation

- Shrinkage methods
  - Ridge regression and the Lasso
  - Dimension reduction
- Labs for each part

Flashback: subset selection

- **Big idea:** if having too many predictors is the problem maybe we can get rid of some

- Three methods:
  - **Best subset:** try all possible combinations of predictors
  - **Forward**: start with no predictors, greedily add one at a time
  - **Backward**: start with all predictors, greedily remove one at a time

Common theme of subset selection:

ultimately, individual predictors are either **IN** or **OUT**

Discussion

- **Question:** what potential problems do you see?
- **Answer:** we're exploring the space of possible models as if there were only finitely many of them, but there are actually infinitely many (why?)

New approach: “regularization”

subset selection

$$Y \approx \beta_0 + \boxed{\cancel{\beta_1}} \cancel{X_1} + \cdots + \boxed{\beta_p} X_p$$

constrain the coefficients

Another way to phrase it:

reward models that **shrink** the coefficient estimates **toward zero**

(and still perform well, of course)

Approach 1: ridge regression

- **Big idea**: minimize RSS plus an additional penalty that rewards small (sum of) coefficient values

Rewards coefficients close to zero

$$\overbrace{\sum_{i=1}^{n}\left(y_i - \beta_0 - \underbrace{\sum_{j=1}^{p}\beta_j x_{ij}}_{\text{Predicted value}}\right)}^{\text{RSS}} + \lambda \overbrace{\sum_{j=1}^{p}\boxed{\beta_j^2}}^{\text{Shrinkage penalty}}$$

Sum over all observations

Observed value

Tuning parameter

Sum over all predictors

* In statistical / linear algebraic parlance, this is an ℓ_2 penalty

Approach 1: ridge regression

- For each value of λ, we only have to fit one model

$$\overbrace{\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)}^{\text{RSS}} + \underset{\text{Tuning parameter}}{\lambda} \overbrace{\sum_{j=1}^{p}\beta_j^2}^{\text{Shrinkage penalty}}$$

- Substantial computational savings over best subset!

Approach 1: ridge regression

- **Question**: what happens when the tuning parameter is **small**?

$$\overbrace{\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)}^{\text{RSS}}+\underset{\substack{\uparrow\\ \text{Tuning}\\ \text{parameter}}}{\lambda}\overbrace{\sum_{j=1}^{p}\beta_j^2}^{\text{Shrinkage penalty}}$$

- **Answer:** just minimizing RSS; simple least-squares

Approach 1: ridge regression

- **Question**: what happens when the tuning parameter is **large**?

$$\overbrace{\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)}^{\text{RSS}} + \underset{\text{Tuning parameter}}{\lambda}\overbrace{\sum_{j=1}^{p}\beta_j^2}^{\text{Shrinkage penalty}}$$

- **Answer:** all coefficients go to zero; turns into null model

Ridge regression: caveat

- RSS is scale-invariant*
- **Question**: is this true of the shrinkage penalty?

$$\overbrace{\sum_{i=1}^{n}\left(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij}\right)}^{\text{RSS}} + \lambda \overbrace{\sum_{j=1}^{p}\beta_j^2}^{\text{Shrinkage penalty}}$$

- **Answer:** no! This means having predictors at different scales would influence our estimate… need to first **standardize** the predictors by dividing by the standard deviation

* multiplying any predictor by a constant doesn't matter

Discussion

- **Question:** why would ridge regression improve the fit over least-squares regression?
- **Answer:** as usual, comes down to **bias-variance tradeoff**
  - As λ increases, flexibility decreases: ↓ variance, ↑ bias
  - As λ decreases, flexibility increases: ↑ variance, ↓ bias
  - **Takeaway:** ridge regression works best in situations where least squares estimates have high variance: trades a small increase in bias for a large reduction in variance

So what's the catch?

- Ridge regression doesn't actually perform variable selection
- Final model will include **all predictors**
  - If all we care about is **prediction accuracy**, this isn't a problem
  - It does, however, pose a challenge for **model interpretation**
- If we want a technique that actually performs variable selection, **what needs to change**?

Approach 2: the lasso

- **(same) Big idea**: minimize RSS plus an additional penalty that rewards small (sum of) coefficient values

$$\overbrace{\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)}^{\text{RSS}} + \underset{\text{Tuning parameter}}{\lambda}\overbrace{\sum_{j=1}^{p}\boxed{|\beta_j|}}^{\text{Shrinkage penalty}}$$

Rewards coefficients close to zero

* In statistical / linear algebraic parlance, this is an ℓ_1 penalty

Discussion

- **Question:** why does that enable us to get coefficients exactly equal to **zero**?

Answer: let's reformulate a bit

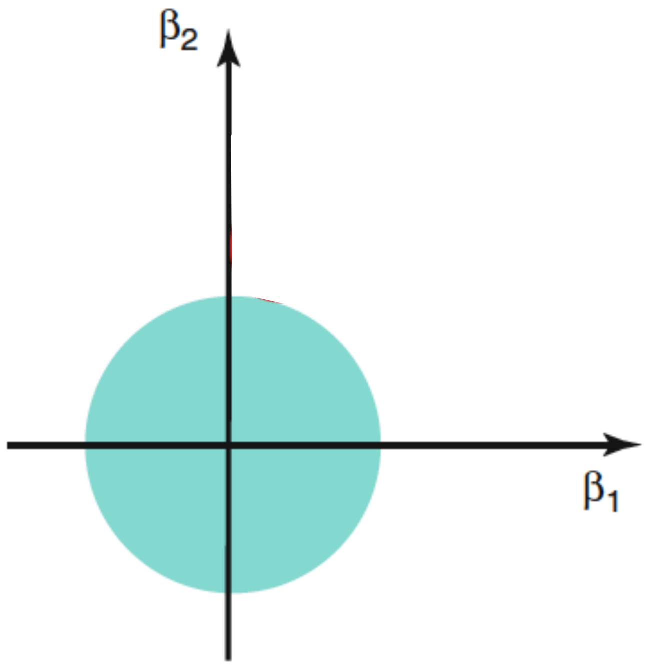
- For each value of λ, there exists a value for s such that:
- Ridge regression:

$$\min_{\beta}(RSS) \text{ subject to } \sum_{j=1}^{p} \beta_j^2 \leq s$$

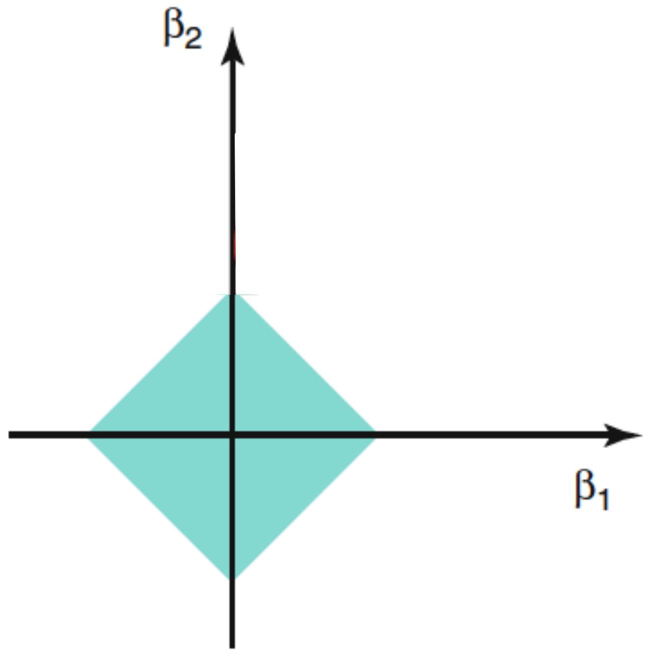
- Lasso:

$$\min_{\beta}(RSS) \text{ subject to } \sum_{j=1}^{p} |\beta_j| \leq s$$

Comparing constraint functions

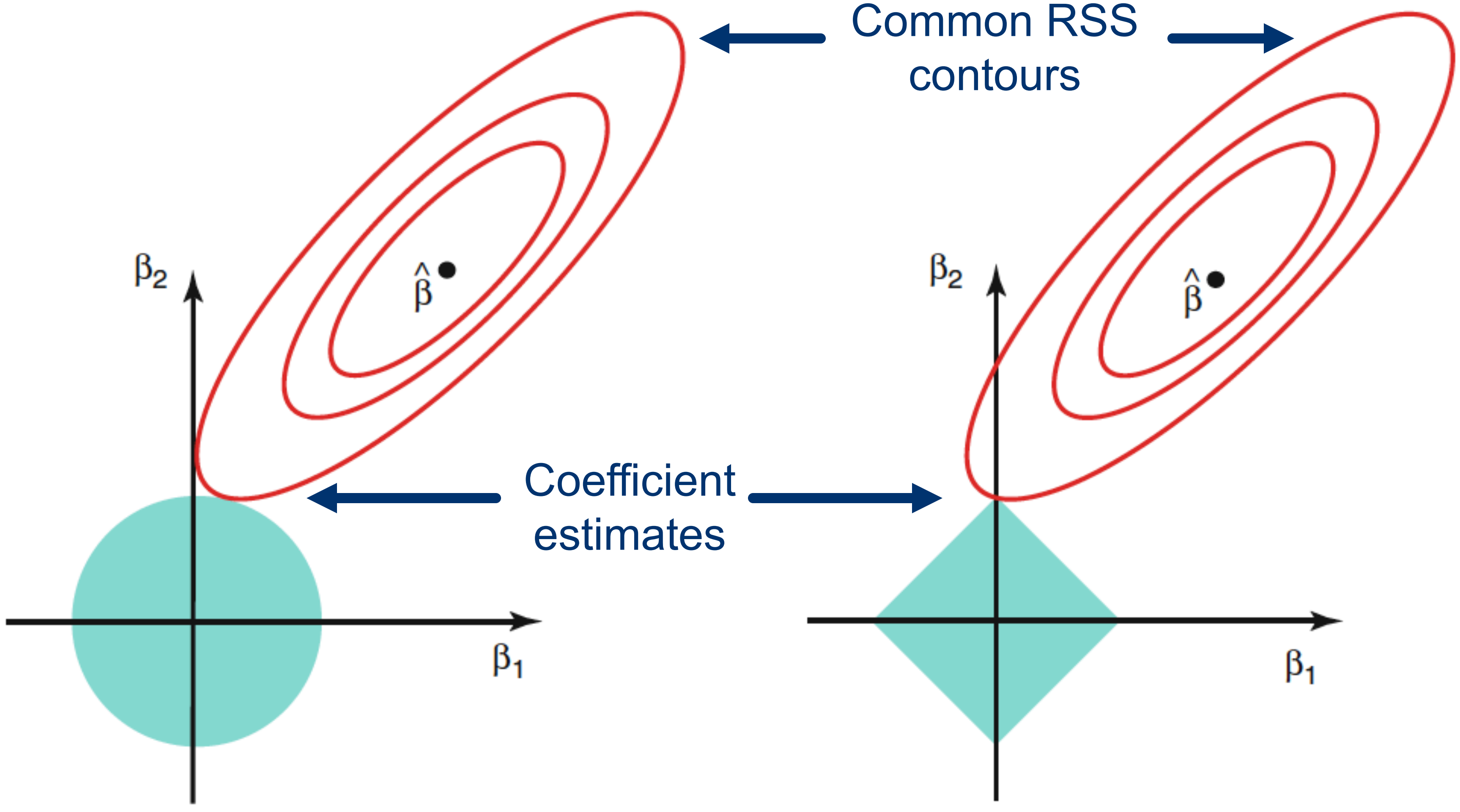


Ridge regression



Lasso

Comparing constraint functions



Ridge regression

Lasso

Comparing ridge regression and the lasso

- Efficient implementations for both (in R and python!)
- Both significantly reduce variance at the expense of a small increase in bias
- **Question**: when would one outperform the other?
- **Answer:**
  - When there are relatively many equally-important predictors, **ridge regression** will dominate
  - When there are small number of important predictors and many others that are not useful, **the lasso** will win

Lingering concern…

- **Question:** how do we choose the right value of λ?
- **Answer:** sweep and cross validate!
  - Because we are only fitting a single model for each λ, we can afford to **try lots of possible values** to find the best ("sweeping")
  - For each λ we test, we'll want to calculate the **cross-validation error** to make sure the performance is consistent

Lab: ridge regression & the lasso

- To do today's lab in R: `glmnet`
- To do today's lab in python: <nothing new>
- Instructions and code:

  [course website]/labs/lab10-r.html

  [course website]/labs/lab10-py.html

- Full version can be found beginning on p. 251 of ISLR

LECTURE 13: DIMENSIONALITY REDUCTION

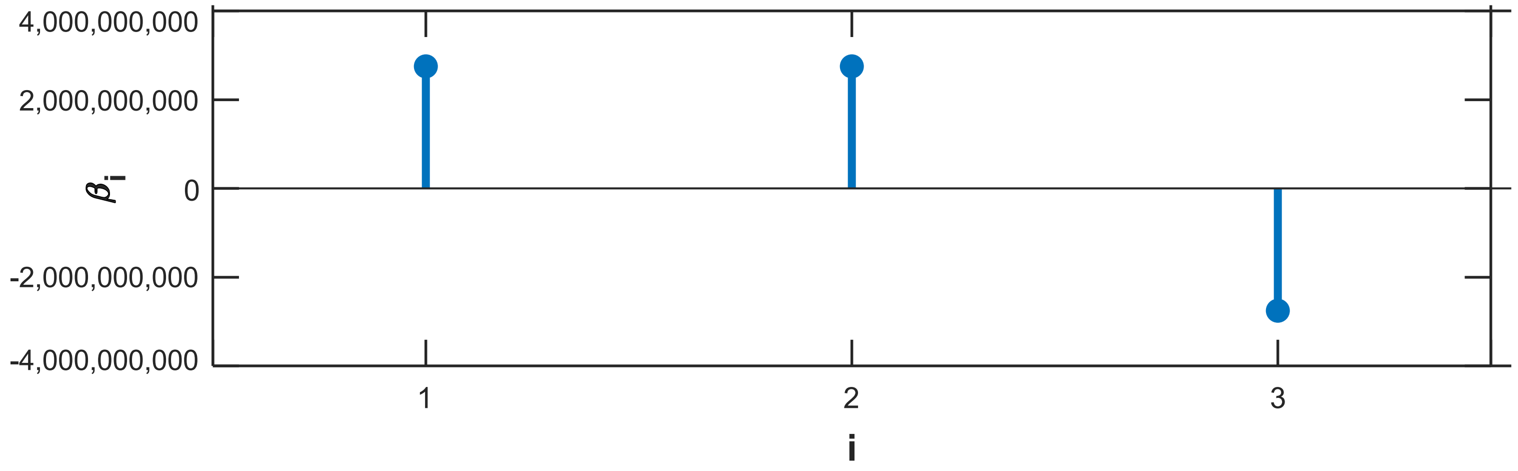
October 25, 2017
SDS 293: Machine Learning

Recap: Ridge Regression and the Lasso

- Both are “shrinkage” methods
- Estimates for the coefficients are ***biased*** toward the origin
  - Biased = “prefers some estimates to others”
  - Does not yield the true value in expectation
- Question: why would we **want** a biased estimate?

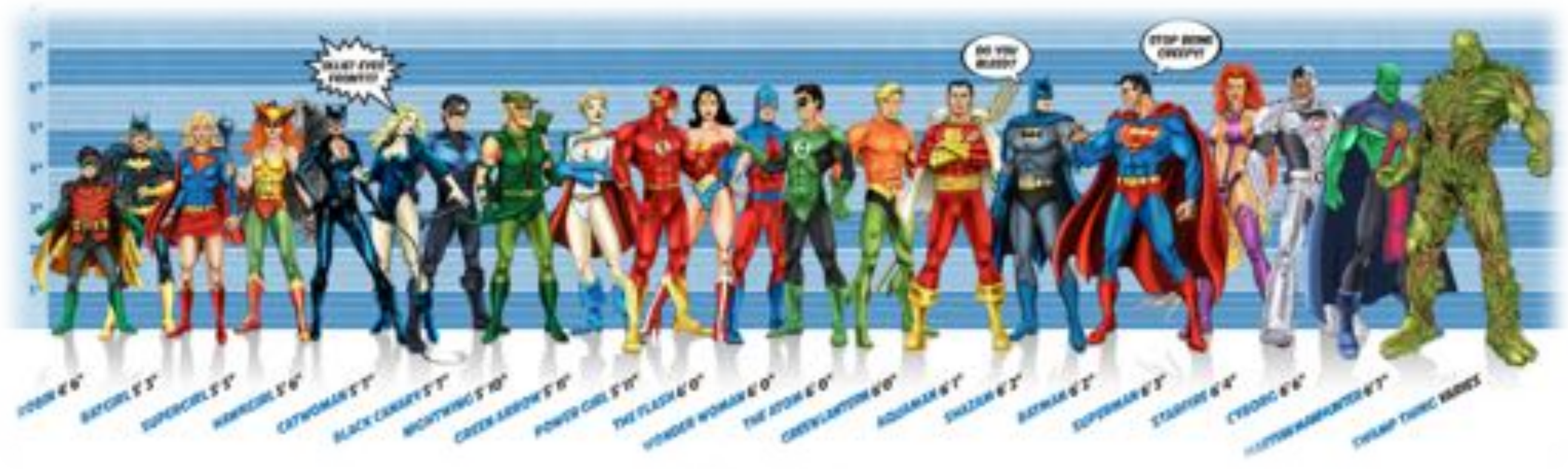
What’s wrong with bias?

- What if your unbiased estimator gives you this?



May want to bias our estimate to **reduce variance**

Flashback: superheroes



$$height = \beta_1\left(\quad \right) + \beta_2\left(\quad \right) + \beta_3\left(\quad \right)$$

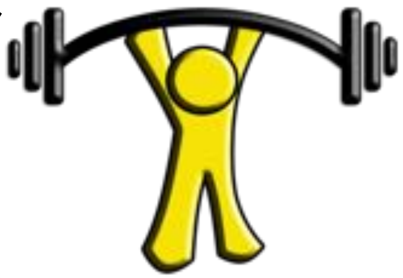
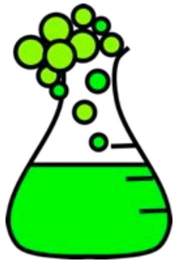
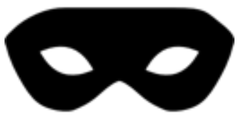
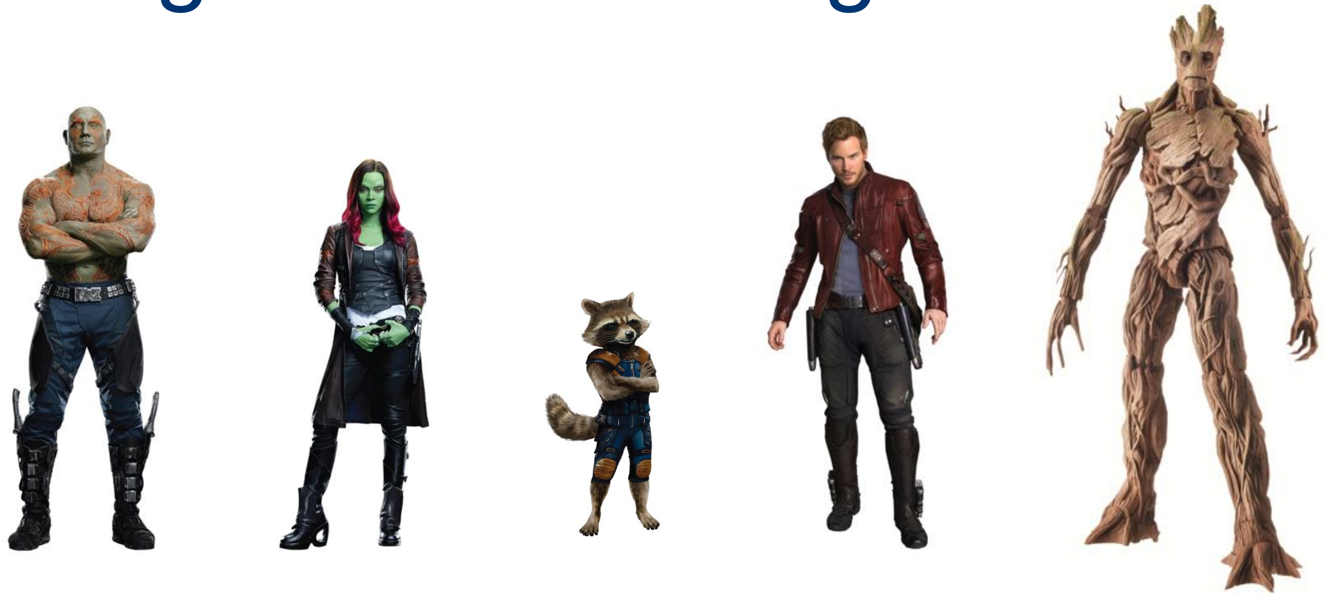




Image credit: Ming Malaykham

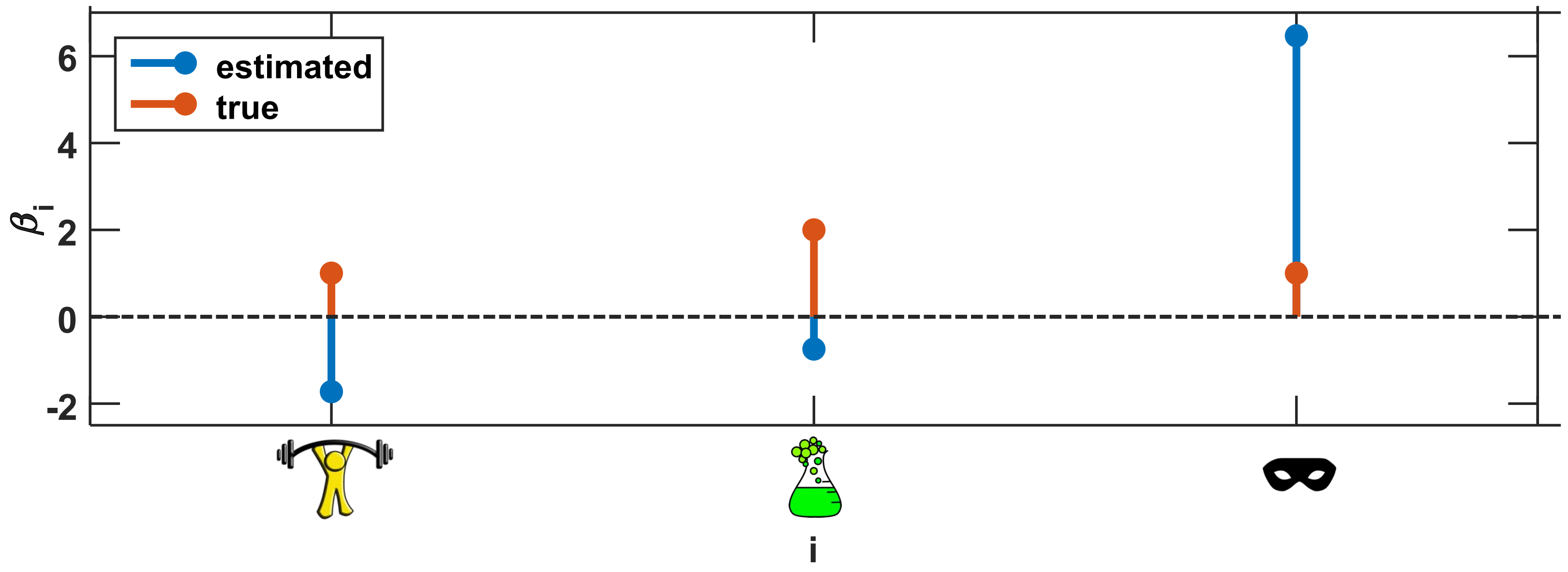
Estimating Guardians' Height



$$\begin{bmatrix} 232.03 \\ 156.29 \\ 113.82 \\ 229.07 \\ 287.72 \end{bmatrix} = 1 \begin{bmatrix} 63.9 \\ 28.9 \\ 54.3 \\ 69.8 \\ 50.4 \end{bmatrix} + 2 \begin{bmatrix} 54.0 \\ 45.1 \\ 13.3 \\ 49.5 \\ 85.4 \end{bmatrix} + 1 \begin{bmatrix} 59.1 \\ 36.9 \\ 33.7 \\ 59.7 \\ 67.9 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \end{bmatrix}$$

Estimate for β

- When we try to estimate using OLS, we get the following:



(Relatively) huge difference between actual and estimated coefficients

What's going on here?

$$\begin{array}{ccccccccc} & & \text{🏋} & & \text{⚗} & & \text{🎭} & & \\ \begin{bmatrix} 232.03 \\ 156.29 \\ 113.82 \\ 229.07 \\ 287.72 \end{bmatrix} & = 1 & \begin{bmatrix} 63.9 \\ 28.9 \\ 54.3 \\ 69.8 \\ 50.4 \end{bmatrix} & + 2 & \begin{bmatrix} 54.0 \\ 45.1 \\ 13.3 \\ 49.5 \\ 85.4 \end{bmatrix} & + 1 & \begin{bmatrix} 59.1 \\ 36.9 \\ 33.7 \\ 59.7 \\ 67.9 \end{bmatrix} & + & \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \end{bmatrix} \end{array}$$

$$\approx avg\left(\text{🏋}, \text{⚗} \right)$$

- Some dimensions are redundant
  - Little information in 3rd dimension not captured by the first two
  - In linear regression, redundancy causes noise to be **amplified**

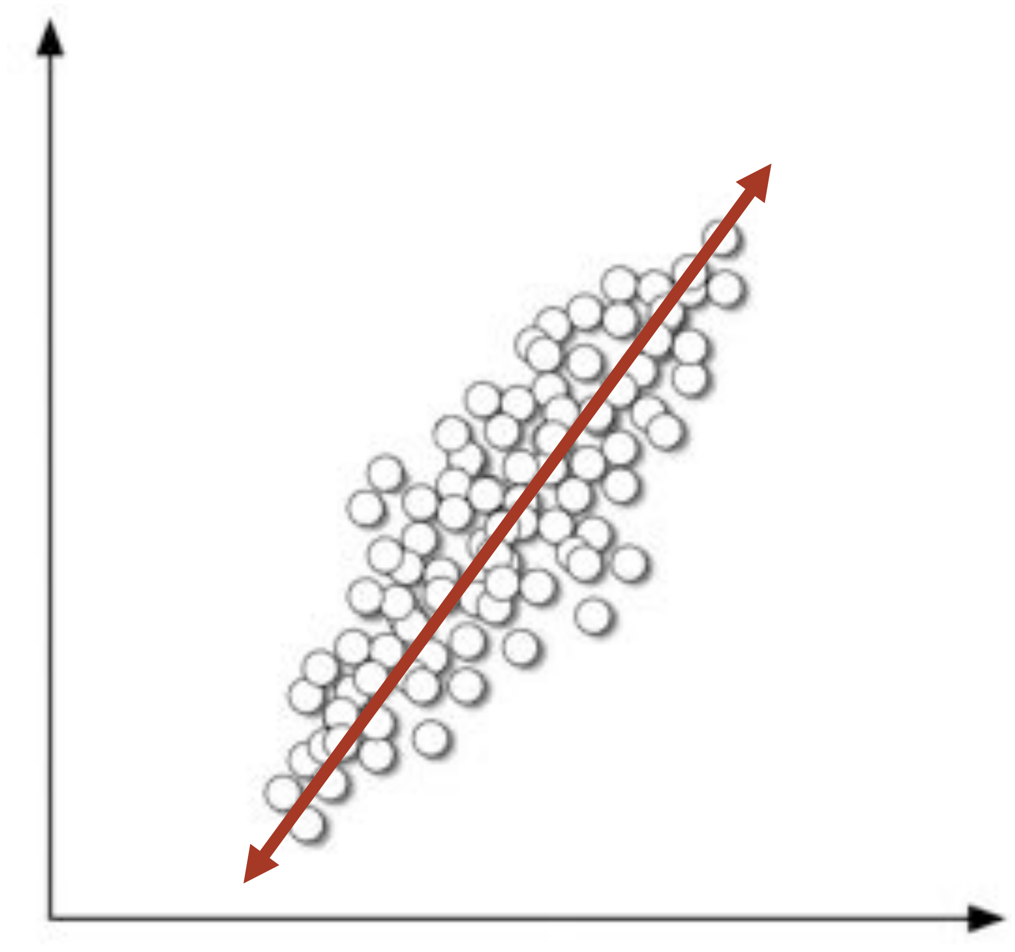
Dimension reduction

- **Current situation**: our data live in *p*-dimensional space, but not all *p* dimensions are equally useful
- **Subset selection**: throw some out
  - Pro: pretty easy to do
  - Con: lose some information
- **Alternate approach**: create **new** features that are combinations of the old ones

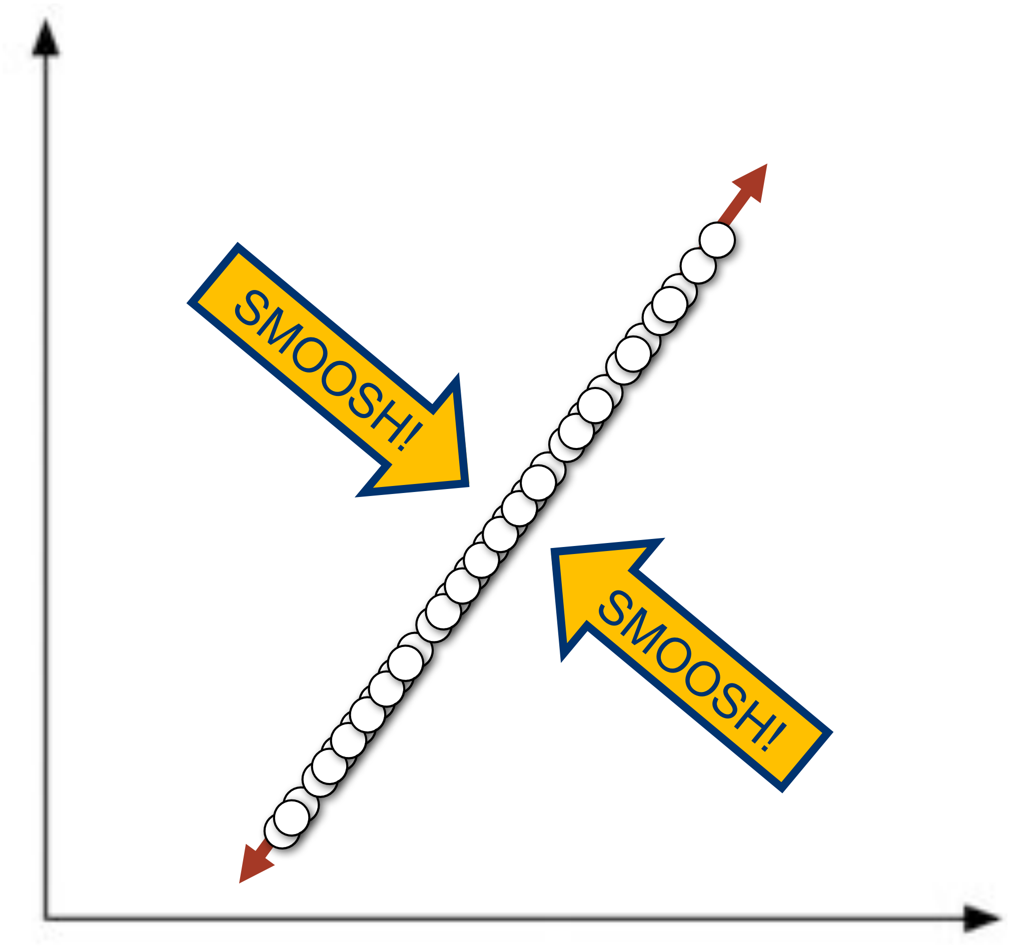
In other words:

Project the data into a new feature space to reduce variance in the estimate

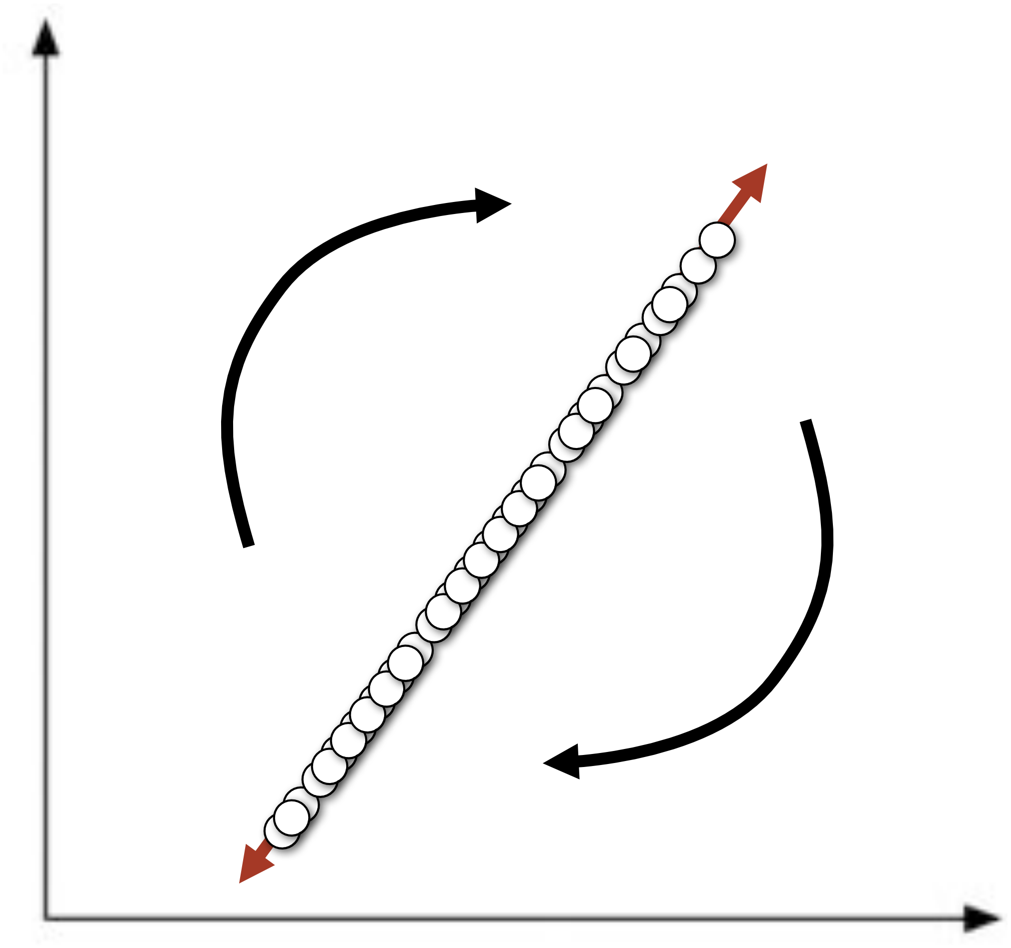
Projection



Projection



Projection



Dimension reduction via projection

- **Big idea**: *transform* the data before performing regression

$$[X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5] \mapsto [Z_1 \quad Z_2]$$

- Then instead of:

$$Y = \beta_0 + \sum_{i=1}^{p} \beta_i X_i + \varepsilon$$

we solve:

$$Y = \theta_0 + \sum_{i=1}^{m} \theta_i Z_i + \varepsilon$$

Linear projection

- New features are **linear combinations** of original data:

$$Z_j = \sum_{i}^{m} \theta_{ij} X_i$$

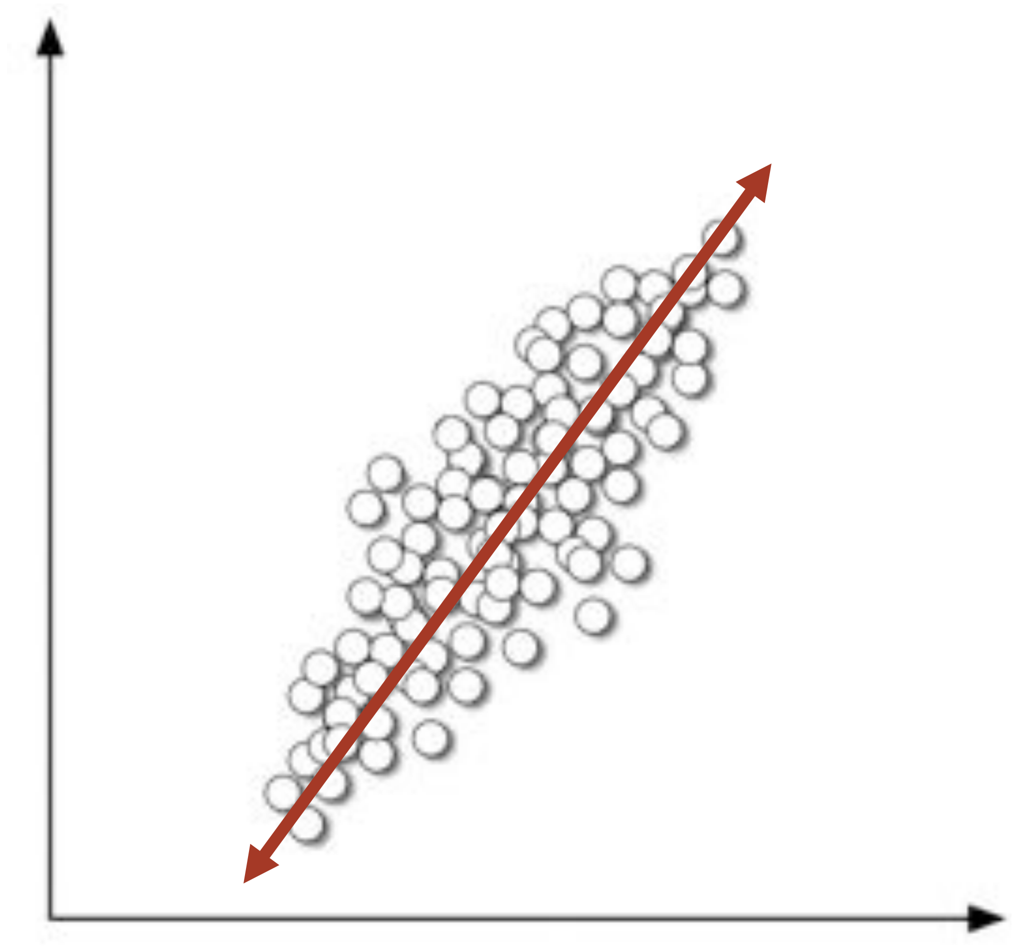
- **MTH211**: multiplying the *data matrix* by a *projection matrix*

$$[Z_1 \quad Z_2] = [X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5] \boxed{\begin{bmatrix} \varphi_{1,1} & \varphi_{1,2} \\ \varphi_{2,1} & \varphi_{2,2} \\ \varphi_{3,1} & \varphi_{3,2} \\ \varphi_{4,1} & \varphi_{4,2} \\ \varphi_{5,1} & \varphi_{5,2} \end{bmatrix}}$$

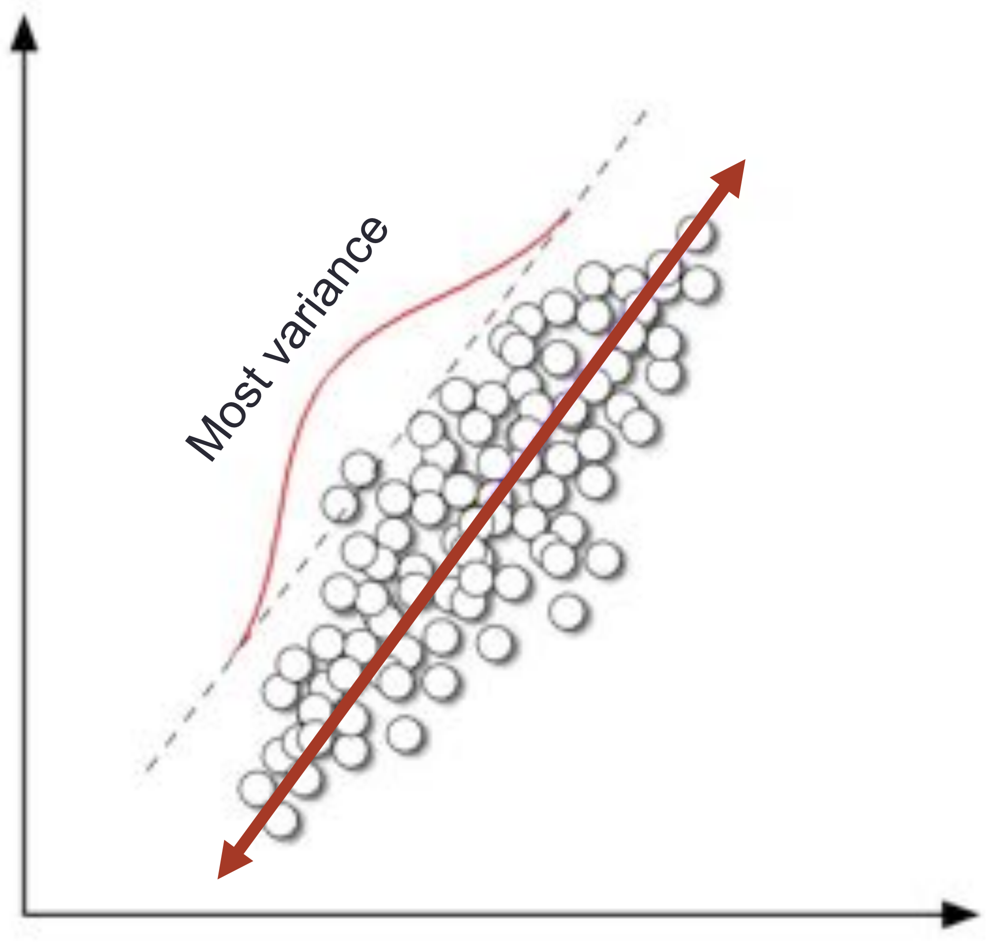
What's the deal with projection?

- Data can be rotated, scaled, and translated without changing the **underlying relationships**
- This means you're allowed to look at the data from whatever angle makes your life easier…

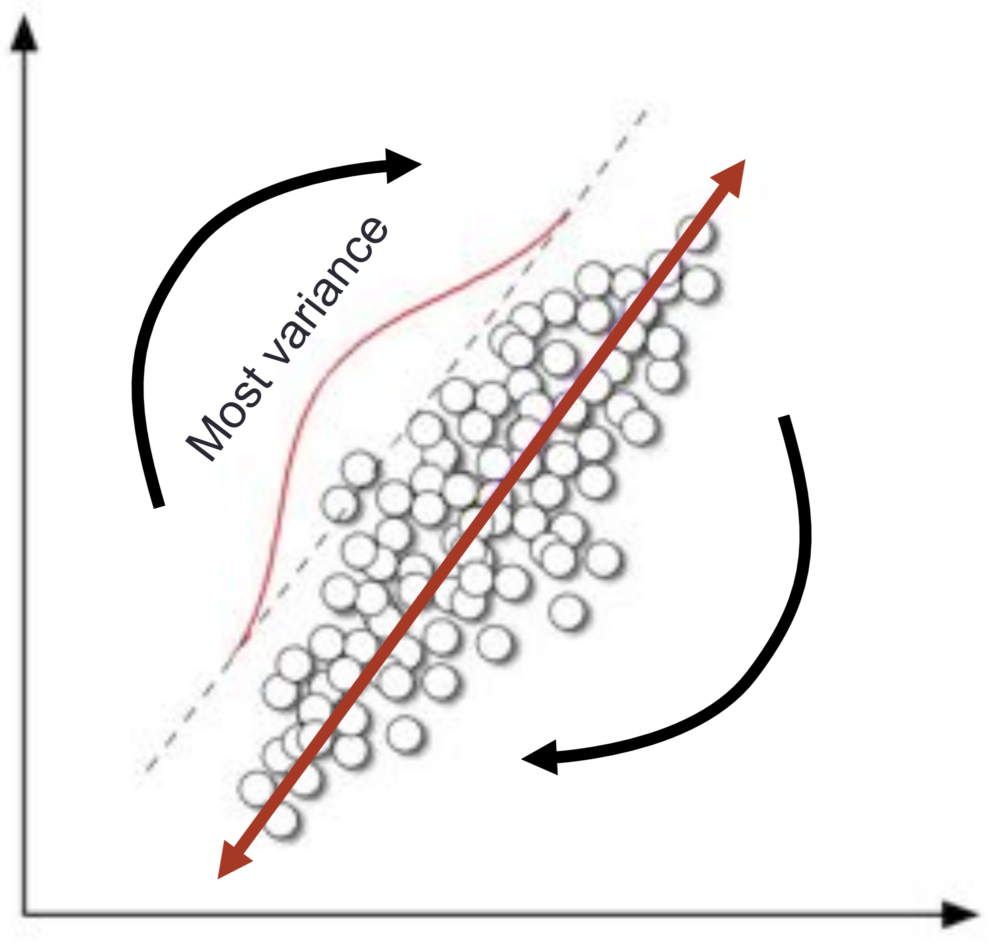
Flashback: why did we pick this line?



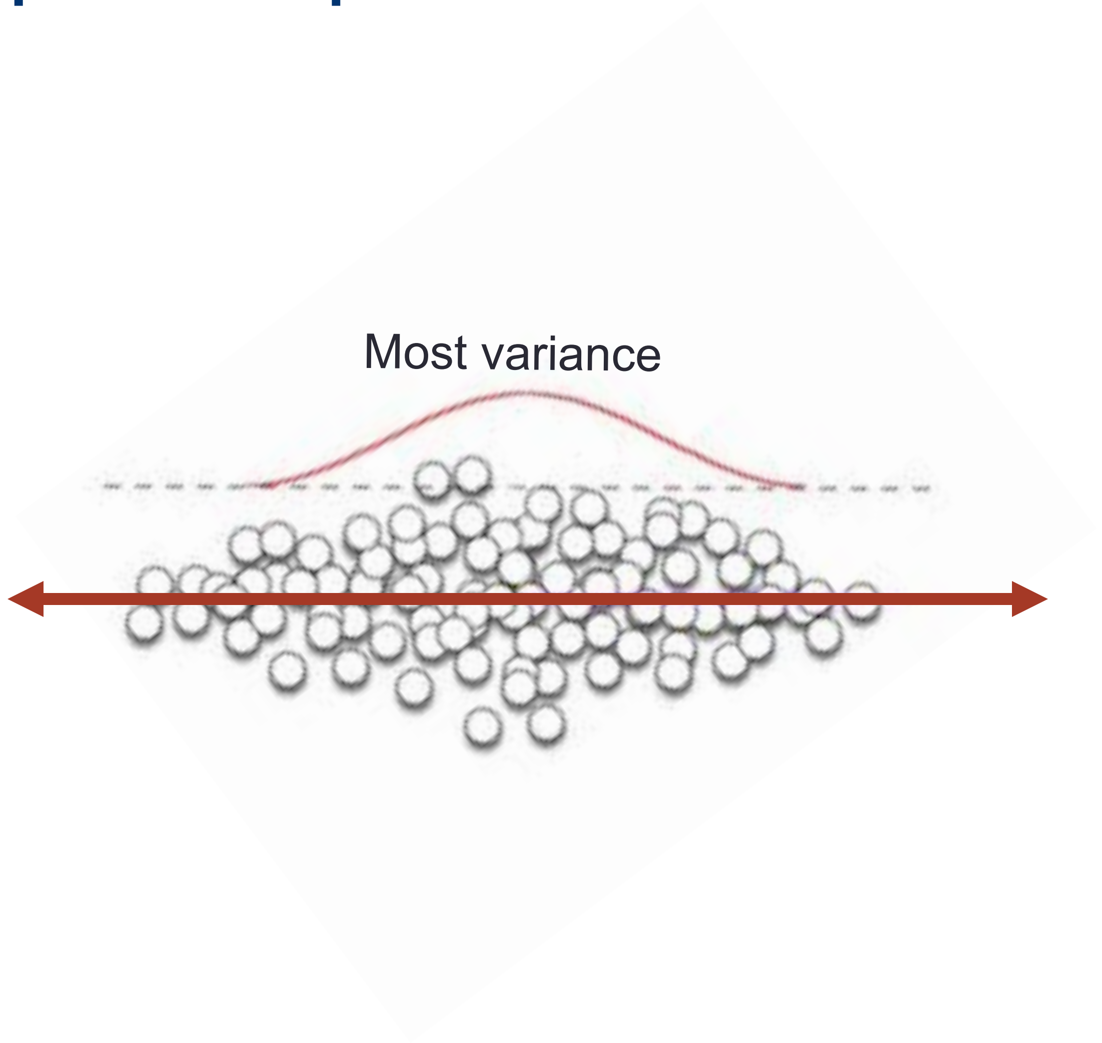
Explains the most **variance** in the data



Imagine this line as a new dimension…



“Principal component”



Mathematically

- The **1st principal component** is the normalized* linear combination of features:

$$Z_1 = \phi_{11}X_1 + \phi_{21}X_2 + \cdots + \phi_{p1}X_p$$

that has the largest variance

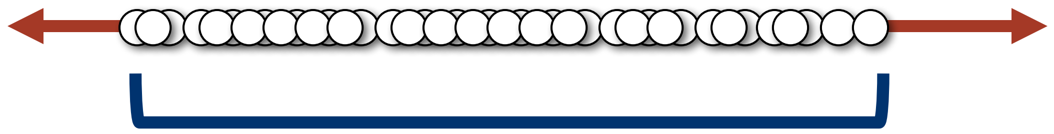
- $\phi_{11}, \dots, \phi_{p1}$: the **loadings** of the 1st principal component

* By **normalized** we mean: $\sum_{j=1}^{p} \phi_{j1}^2 = 1$

Using loadings to project

Multiply by loading vector to project (“smoosh”) each observation onto the line:

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip}$$



These values are called the **scores** of the 1st principal component

Additional principal components

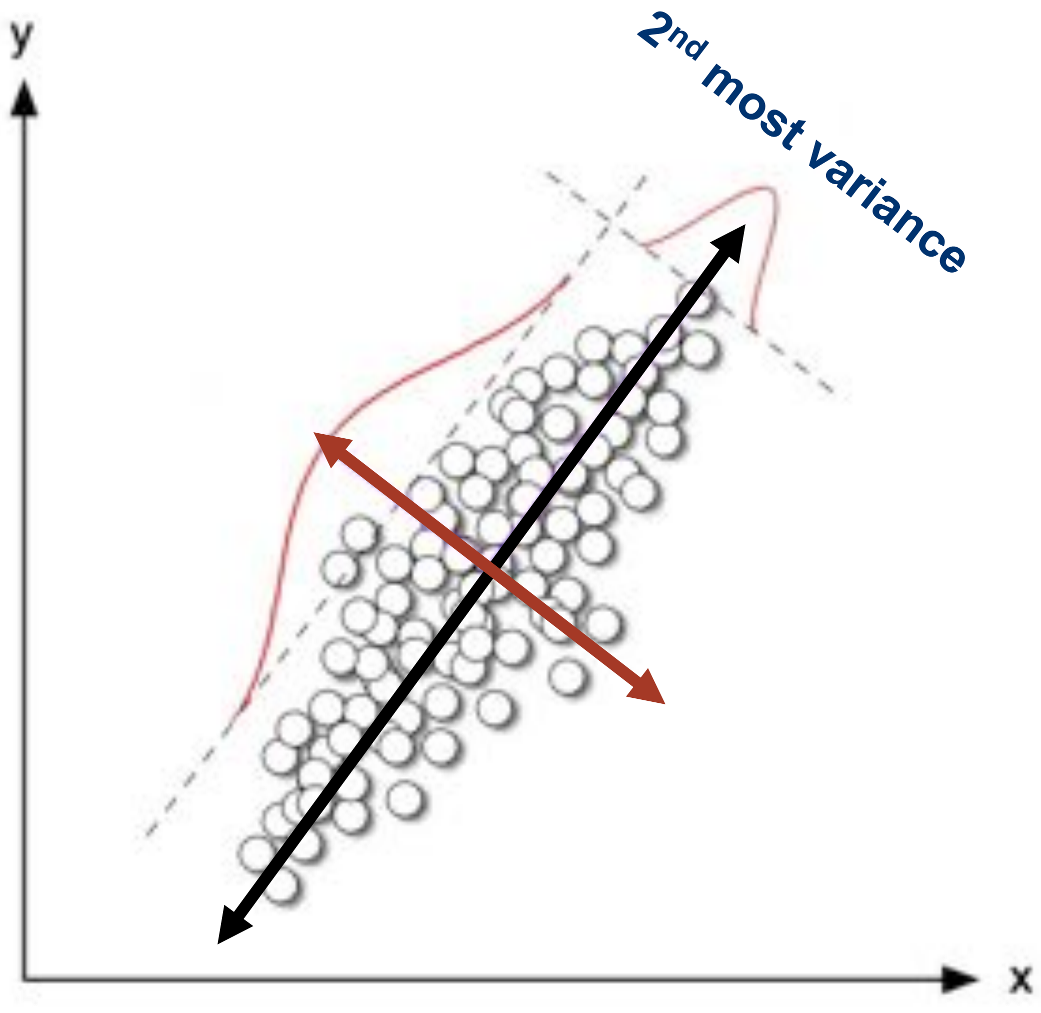
- **2nd principal component** is the normalized linear combination of the features

$$Z_2 = \phi_{12}X_1 + \phi_{22}X_2 + \cdots + \phi_{p2}X_p$$

that has maximal variance out of all linear combinations that are **uncorrelated** with Z_1 (why does that matter?)

- Fun fact:

Principal components are orthogonal



Generating additional principal components

- We can think of this recursively
- To find the M^{th} principal component . . .
  - Find the first $(M - 1)$ principal components
  - Subtract the projection into that space
  - Maximize the variance in the remaining *complementary* space

Regression in the principal components

- **Original objective**: solve for β in

$$Y = \beta_0 + \sum_{i}^{p} \beta_i X_i + \varepsilon$$

(that's still our goal)

- Now we're going to work in the new feature space:

$$Y = \theta_0 + \sum_{i}^{M} \theta_i Z_i + \varepsilon$$

Regression in the principal components

- *Remember*: the new features are **related** to the old ones:

$$Z_j = \sum_{i=1}^{p} \phi_{ij} X_i$$

- So we're computing:

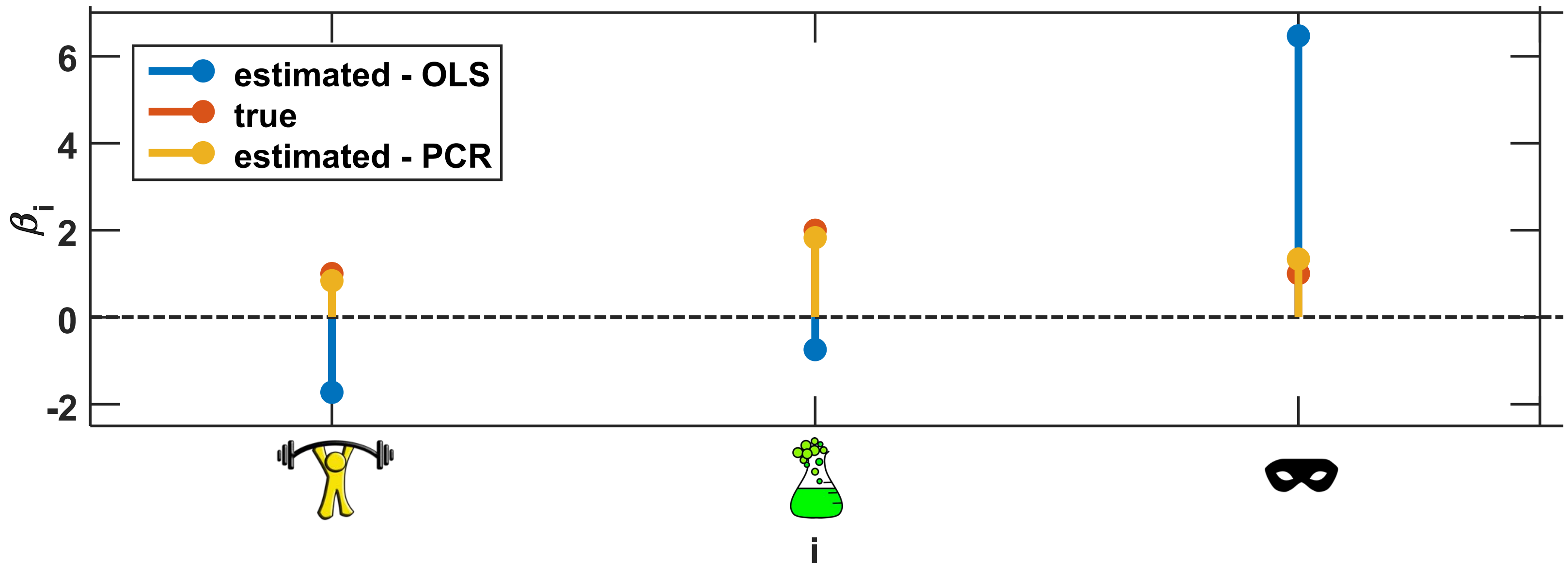
$$\begin{aligned} Y &= \theta_0 + \sum_{j=1}^{M} \theta_j Z_j + \varepsilon \\ &= \theta_0 + \boxed{\sum_{j=1}^{M} \theta_j \sum_{i=1}^{p} \phi_{ij} X_i} + \varepsilon \\ \mapsto \beta_i &= \sum_{j=1}^{M} \theta_j \phi_{ij} \end{aligned}$$

Back to the Guardians

$$\begin{bmatrix} 232.03 \\ 156.29 \\ 113.82 \\ 229.07 \\ 287.72 \end{bmatrix} = 1\begin{bmatrix} 63.9 \\ 28.9 \\ 54.3 \\ 69.8 \\ 50.4 \end{bmatrix} + 2\begin{bmatrix} 54.0 \\ 45.1 \\ 13.3 \\ 49.5 \\ 85.4 \end{bmatrix} + 1\begin{bmatrix} 59.1 \\ 36.9 \\ 33.7 \\ 59.7 \\ 67.9 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \end{bmatrix}$$

Back to the Guardians

- What happens if we use 2 components instead of 3?



Using only the principal components significantly improves our estimate!

Comparison with ridge regression and the lasso

- What similarities do you see?
  - Reduces dimensionality of the solution space (like Lasso)
  - Finds a solution in the space of all features (like RR)
  - Results can be difficult to interpret (like RR)

Problems with PCR

- We selected principal components based on predictors (not what we're trying to predict!)
- This could be problematic (why?)
  - What if the values you're trying to predict aren't correlated with the first few components?
  - You lose all predictive power!

Partial least squares (PLS)

- A *supervised* form of PCR
- Feature derivation algorithm is similar:
  - Find the (*M*-1) ~~principal~~ **most correlated** components
  - Subtract the projection into that space
  - Maximize the ~~variance~~ **correlation with the response** in the remaining *complementary* space
- As before, we perform least squares on the new features
- We still use the formulation

$$Z_j = \sum_{i=1}^{p} \phi_{ij} X_i$$

- But now ϕ is computed by applying linear regression to *each* predictor

Wrapping up: PCR/PLS comparison

- Both derive a small number of orthogonal predictors for linear regression
- PCR is more biased
  - Emphasizes stability at the expense of versatility
- PLS estimates have higher variance
  - May build new features that aren’t as stable
  - But high variance is better than infinite variance

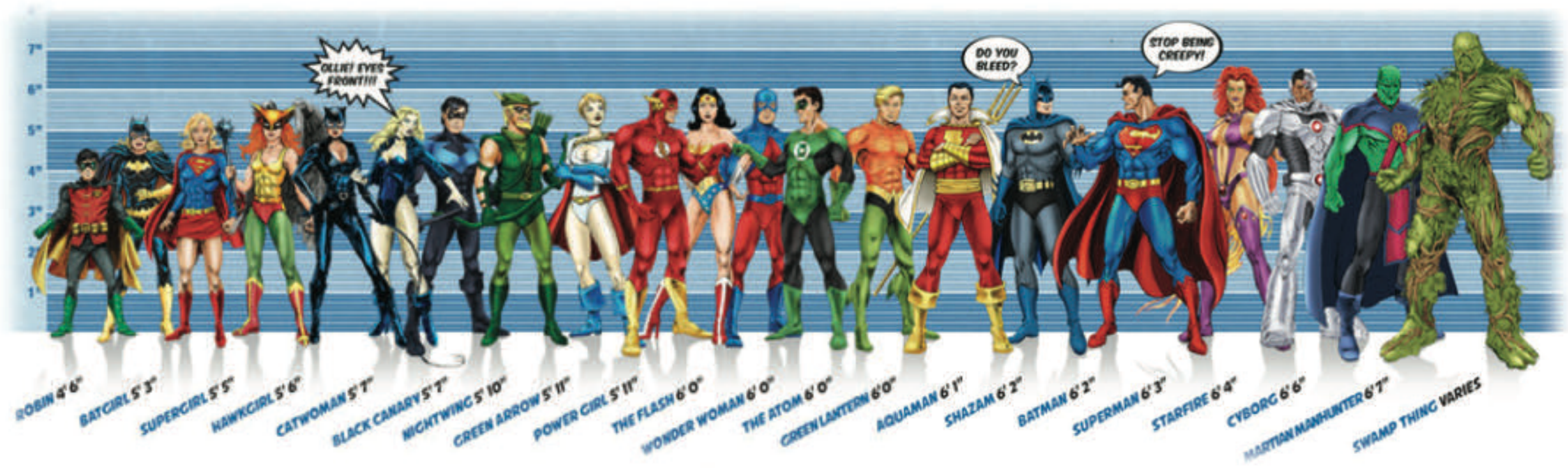
Lab: PCR and PLS

- To do today's lab in R: **`pls`**
- To do today's lab in python: <nothing new>
- Instructions and code:

  [course website]/labs/lab11-r.html

  [course website]/labs/lab11-py.html
- Full version can be found beginning on p. 256 of ISLR

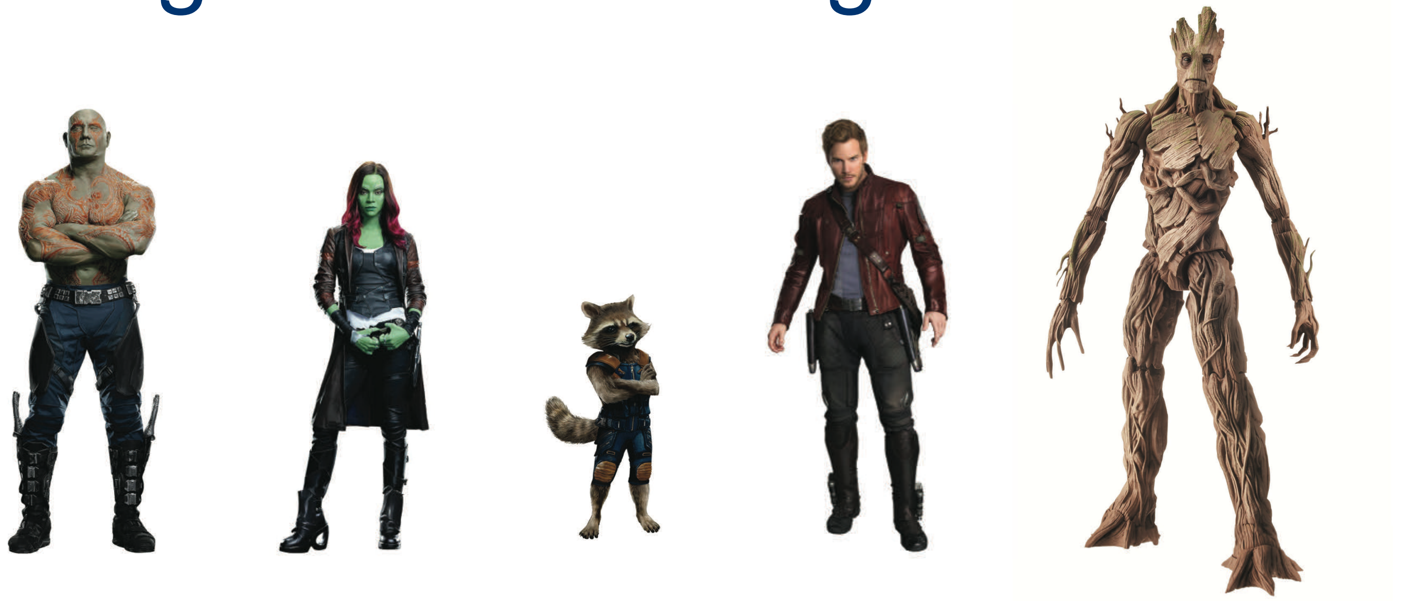
Flashback: superheroes



$$height = \beta_1 \left(\quad \right) + \beta_2 \left(\quad \right) + \beta_3 \left(\quad \right)$$

Image credit: Ming Malaykham

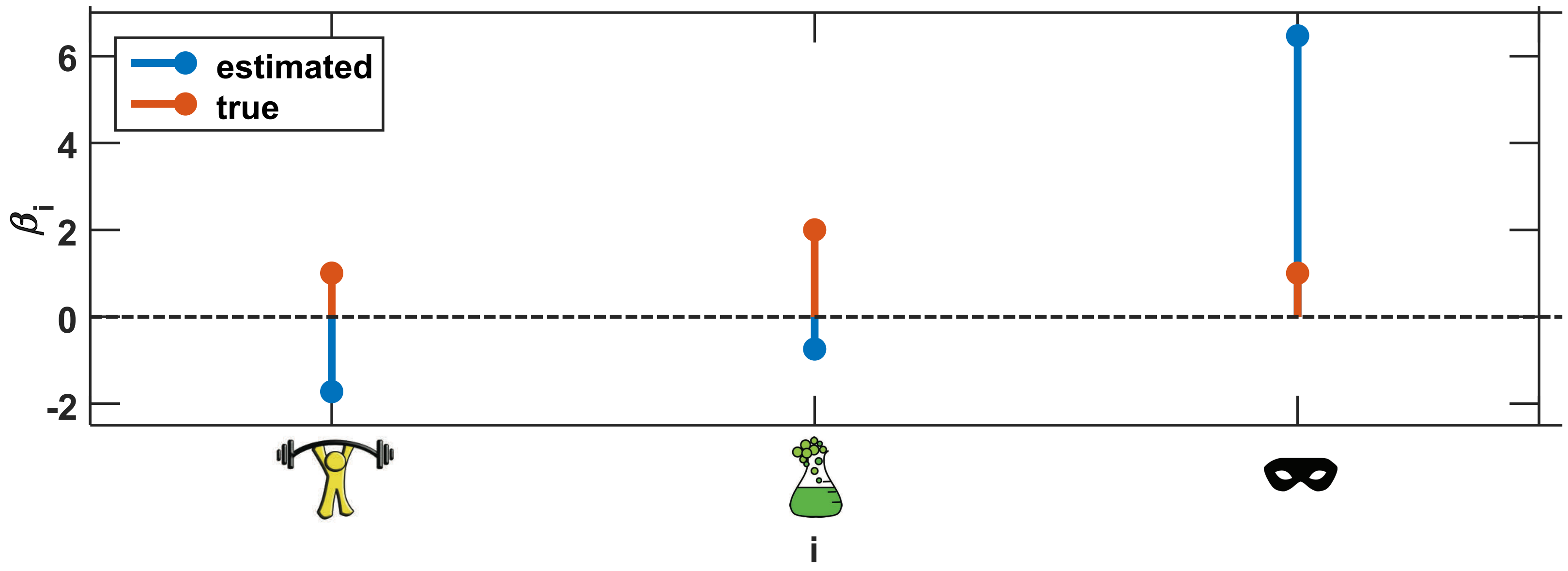
Estimating Guardians' Height



$$\begin{bmatrix} 232.03 \\ 156.29 \\ 113.82 \\ 229.07 \\ 287.72 \end{bmatrix} = 1\begin{bmatrix} 63.9 \\ 28.9 \\ 54.3 \\ 69.8 \\ 50.4 \end{bmatrix} + 2\begin{bmatrix} 54.0 \\ 45.1 \\ 13.3 \\ 49.5 \\ 85.4 \end{bmatrix} + 1\begin{bmatrix} 59.1 \\ 36.9 \\ 33.7 \\ 59.7 \\ 67.9 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \end{bmatrix}$$

Estimate for β

- When we try to estimate using OLS, we get the following:



(Relatively) huge difference between actual and estimated coefficients

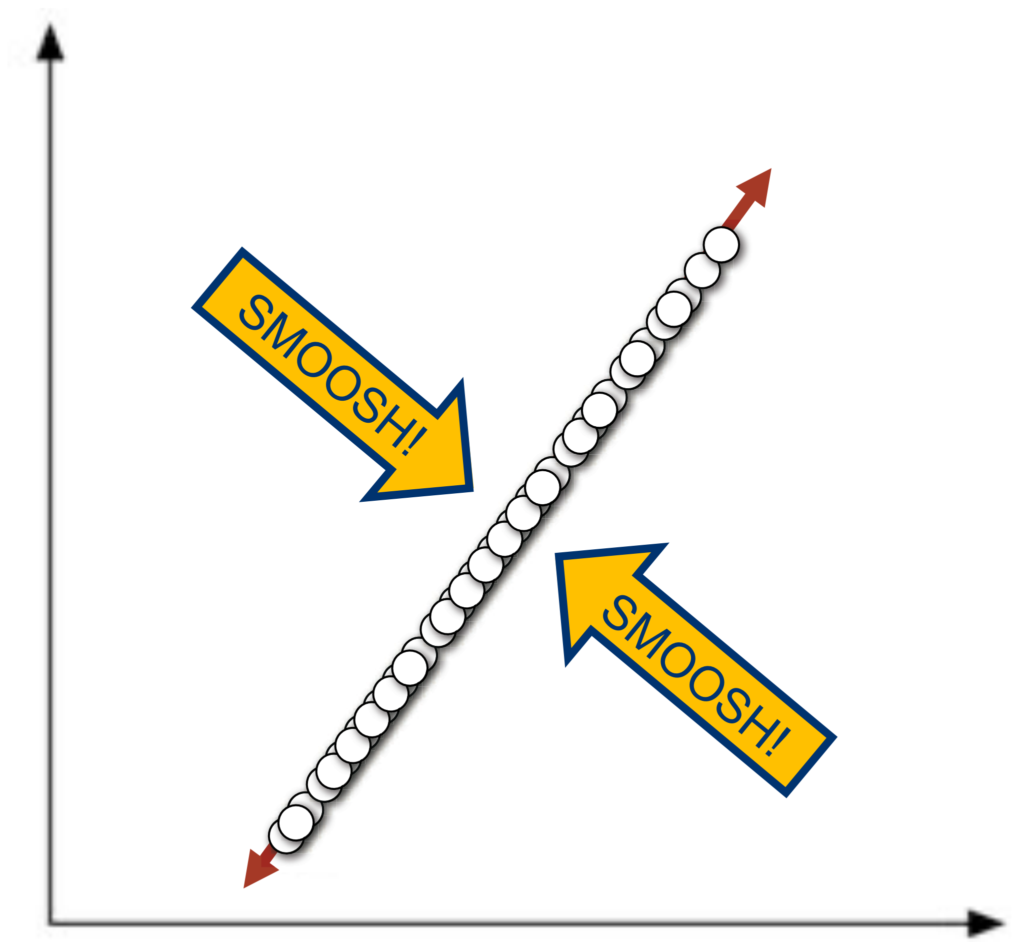
What’s going on here?

$$\begin{bmatrix}232.03\\156.29\\113.82\\229.07\\287.72\end{bmatrix} = 1\overset{\text{🏋}}{\begin{bmatrix}63.9\\28.9\\54.3\\69.8\\50.4\end{bmatrix}} + 2\overset{\text{🧪}}{\begin{bmatrix}54.0\\45.1\\13.3\\49.5\\85.4\end{bmatrix}} + 1\overset{\text{🎭}}{\boxed{\begin{bmatrix}59.1\\36.9\\33.7\\59.7\\67.9\end{bmatrix}}} + \begin{bmatrix}\varepsilon_1\\\varepsilon_2\\\varepsilon_3\\\varepsilon_4\\\varepsilon_5\end{bmatrix}$$

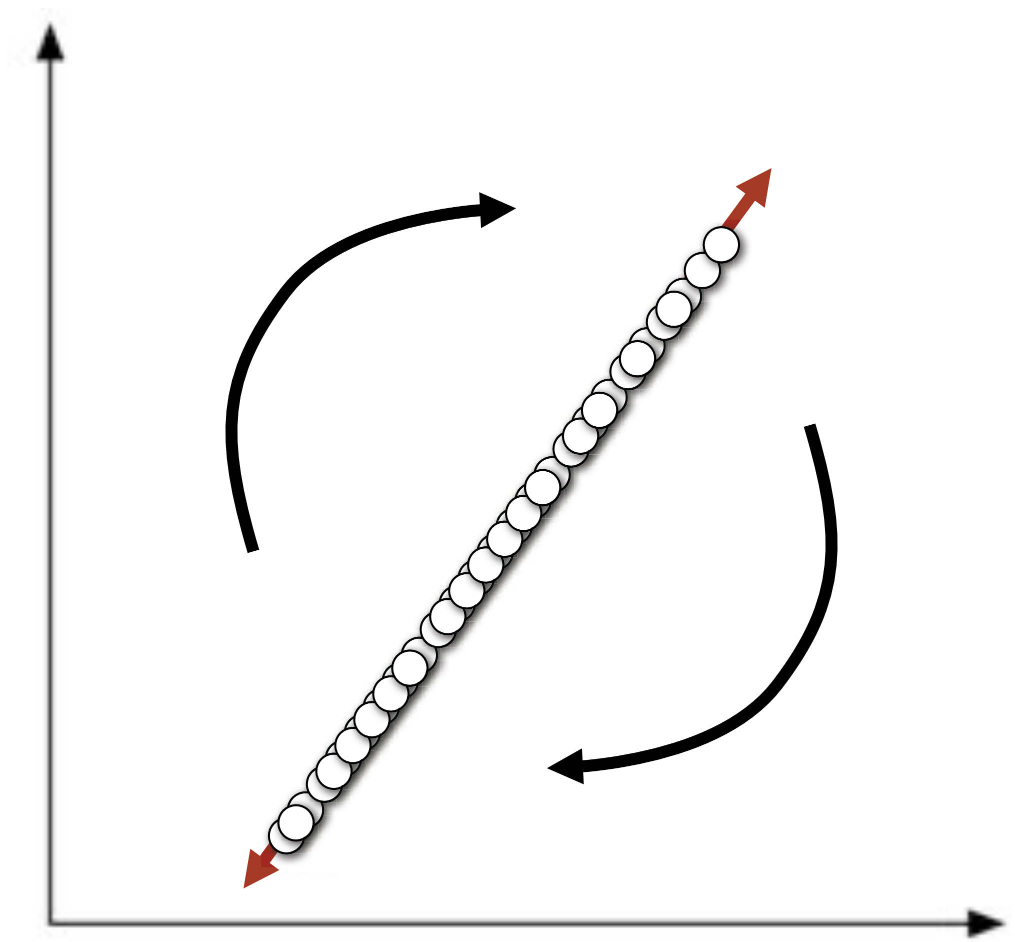
$$\approx avg\left(\text{🏋}, \text{🧪}\right)$$

- Some dimensions are redundant
  - Little information in 3rd dimension not captured by the first two
  - In linear regression, redundancy causes noise to be **amplified**

Projection



Projection



Linear projection

- New features are **linear combinations** of original data:

$$Z_j = \sum_{i}^{m} \theta_{ij} X_i$$

- **MTH211**: multiplying the *data matrix* by a *projection matrix*

$$[Z_1 \quad Z_2] = [X_1 \quad X_2 \quad X_3 \quad X_4 \quad X_5] \begin{bmatrix} \varphi_{1,1} & \varphi_{1,2} \\ \varphi_{2,1} & \varphi_{2,2} \\ \varphi_{3,1} & \varphi_{3,2} \\ \varphi_{4,1} & \varphi_{4,2} \\ \varphi_{5,1} & \varphi_{5,2} \end{bmatrix}$$

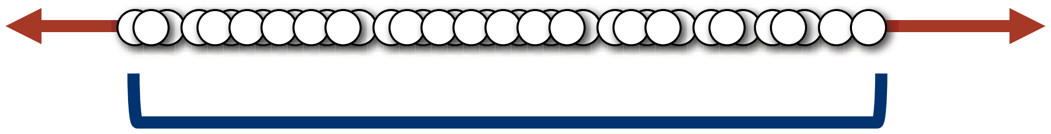
What's the deal with projection?

- Data can be rotated, scaled, and translated without changing the **underlying relationships**
- This means you're allowed to look at the data from whatever angle makes your life easier…

Using loadings to project

Multiply by loading vector to project (“smoosh”) each observation onto the line:

$$z_{i1} = \phi_{11}x_{i1} + \phi_{21}x_{i2} + \cdots + \phi_{p1}x_{ip}$$



These values are called the **scores** of the 1st principal component

Regression in the principal components

- *Remember*: the new features are **related** to the old ones:

$$Z_j = \sum_{i=1}^{p} \phi_{ij} X_i$$

- So we're computing:

$$Y = \theta_0 + \sum_{j=1}^{M} \theta_j Z_j + \varepsilon$$

$$= \theta_0 + \sum_{j=1}^{M} \theta_j \sum_{i=1}^{p} \phi_{ij} X_i + \varepsilon$$

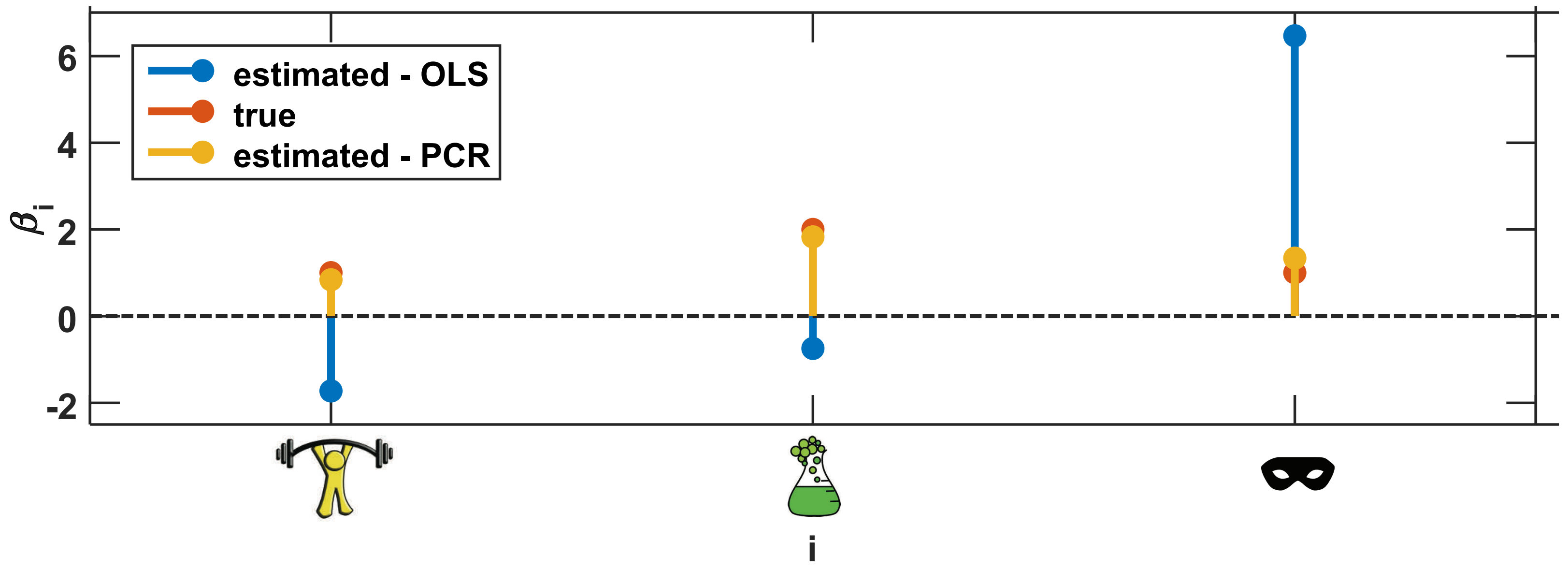
$$\mapsto \beta_i = \sum_{j=1}^{M} \theta_j \phi_{ij}$$

Back to the Guardians

$$\begin{bmatrix} 232.03 \\ 156.29 \\ 113.82 \\ 229.07 \\ 287.72 \end{bmatrix} = 1 \begin{bmatrix} 63.9 \\ 28.9 \\ 54.3 \\ 69.8 \\ 50.4 \end{bmatrix} + 2 \begin{bmatrix} 54.0 \\ 45.1 \\ 13.3 \\ 49.5 \\ 85.4 \end{bmatrix} + 1 \begin{bmatrix} 59.1 \\ 36.9 \\ 33.7 \\ 59.7 \\ 67.9 \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \varepsilon_4 \\ \varepsilon_5 \end{bmatrix}$$

Back to the Guardians

- What happens if we use 2 components instead of 3?



Using only the principal components significantly improves our estimate!

Comparison with ridge regression and the lasso

- What similarities do you see?
  - Reduces dimensionality of the solution space (like Lasso)
  - Finds a solution in the space of all features (like RR)
  - Results can be difficult to interpret (like RR)

Problems with PCR

- We selected principal components based on predictors (not what we're trying to predict!)
- This could be problematic (why?)
  - What if the values you're trying to predict aren't correlated with the first few components?
  - You lose all predictive power!