Feature Engineering

SYS 6018 | Spring 2023

feature-engr.pdf

Contents

1 Feature Engineering

Feature Engineering and Selection: A Practical Approach for Predictive Models by Max Kuhn and Kjell Johnson

> ... we are sometimes frustrated to find that the best models have less-than-anticipated, less-than-useful useful predictive performance. This lack of performance may be due to a simple to explain, but difficult to pinpoint, cause: **relevant predictors that were collected are represented in a way that models have trouble achieving good performance**.
>
> Key relationships that are not directly available as predictors may be between the response and:
>
> - a transformation of a predictor,
> - an interaction of two or more predictors such as a product or ratio,
> - a functional relationship among predictors, or
> - an equivalent re-representation of a predictor.
>
> Adjusting and reworking the predictors to enable models to better uncover predictor-response relationships has been termed *feature engineering*.

2 Feature Transformations

In this section, we are only dealing with transforming *individual features*. Transforming many features together (e.g., PCA) will be addressed in the dimension reduction/expansion section.

Also, keep in mind that we can transform a predictor variable and keep both the new and original features in the model.

2.1 Categorical Features

2.1.1 Nominal (unordered)

- For nominal (unordered) features, binary (dummy) encoding is common. This creates one binary column per level (one-hot) or chooses one level to be the baseline and creates 1-# levels new columns (this is sometimes called dummy encoding).
  - The one-hot encoding may cause computational issues if the model matrix is overdetermined (e.g., $(X^\mathsf{T}X)^{-1}$ isn't invertible). But no problem for elastic net models.
- Dummy encoding creates additional predictors (degrees of freedom) which can inflate the variance (even with lasso/enet).
  - Grouping the rare levels into an "Other" category can help, but at the risk of masking real effects
- Another problem occurs when a new level appears only the test data. A model won't know what prediction to make for the unseen value.
  - This comes up for rare levels in resampling (cross-validation)
- Supervised approaches can also be used. Consider encoding a level with the mean outcome for that level.
  - Use out-of-sample data for encoding to prevent leakage
  - see CatBoost for interesting approach

The data have a nominal feature and numeric outcome. Here's a sample of 10 rows:

nominal	outcome	mean_encoding
D	1.19	1.36
C	1.03	0.20
B	0.89	2.96
C	-1.02	0.20
C	2.38	0.20
D	-1.32	1.36
B	2.37	2.96
C	-0.51	0.20
B	6.20	2.96
D	2.41	1.36

Overall, these are the group means:

nominal	mu
A	1.92
B	2.96
C	0.20
D	1.36

which is where the `mean_encoding` values come from.

- Some tree-based models can handle categorical predictors without need for dummy encoding. You'll recall this often leads to too many splits on the categorical predictors. But for some data this works better (http://www.feat.engineering/categorical-trees.html).

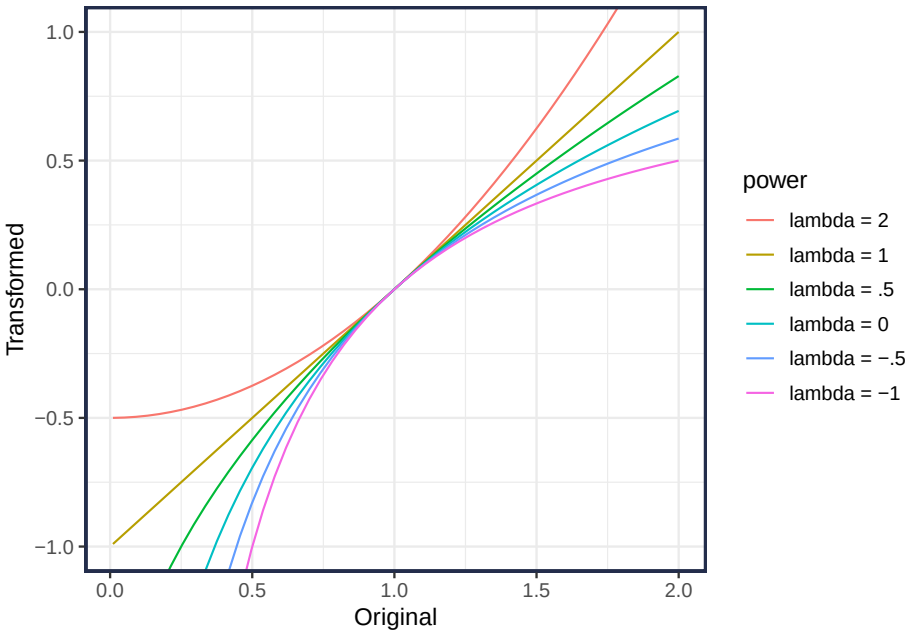
2.1.2 Ordinal

- Some categorical featured may be ordered but not necessarily on an interval scale (e.g., Likert)
- One option is to encode the values onto the integers (e.g., worst = 1, best = 5) and treat as numeric.
  - For models that can include non-linear components (e.g., trees), this can work well
  - For linear models, *polynomial contrasts* can be used to capture non-linear effects.
- One option is to ignore the ordering and treat as nominal (e.g., and dummy encode)
  - But this can contribute to overfitting (high variance) since extra edf needed to capture the ordered effects (if it exists)

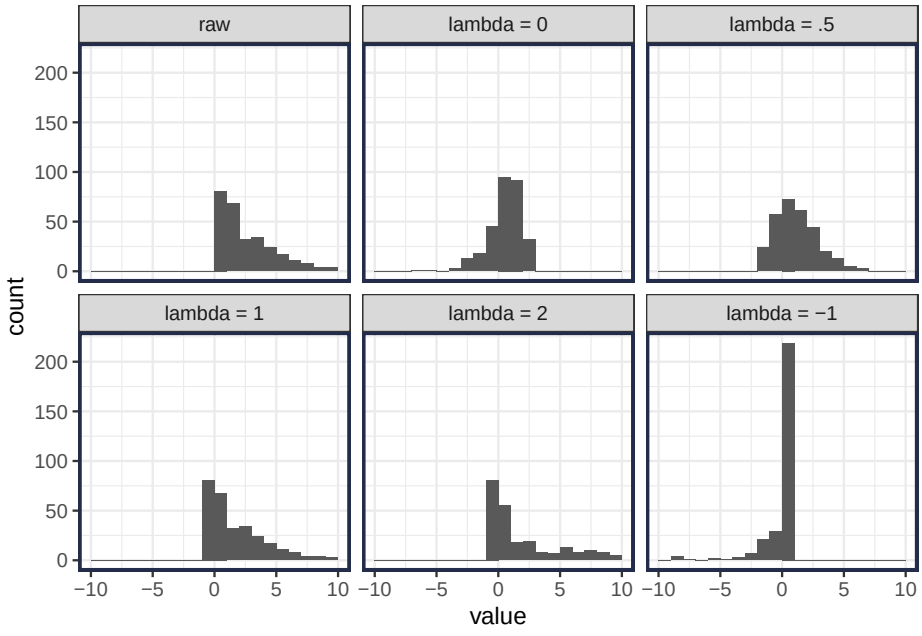
2.2 Numeric Features

- Standard mathematical operations (log, sqrt, exp, inverse)
- Power Transformations like Box-Cox, Yeo-Johnson. Commonly used to transform data to more normal/Gaussian empirical distribution.
- Box-Cox transformation works for positive data and has parameter λ

$$x' = \begin{cases} (x^\lambda - 1)/\lambda & \text{if } \lambda \neq 0 \\ \log(x) & \text{if } \lambda = 0 \end{cases}$$



- $\lambda = 0$ gives log transformation ($\log(x)$)
- $\lambda = 1$ is untransformed ($x - 1$)
- $\lambda = 1/2$ is square root ($2(\sqrt{x} - 1)$)
- $\lambda = 2$ is squared ($(x^2 - 1)/2$)
- $\lambda = -1$ is negative inverse ($1 - 1/x$)
- An optimal value of λ can be found to e.g., make data more symmetric, more linear relationship



- Reminders:
  - OLS will work best when predictors are linearly related to the outcome
  - In OLS, the **residuals** should have symmetric distribution
  - In general, predictors don't need to be normally distributed

- But LDA/QDA will work best when they are *conditionally* Gaussian

2.2.1 Scaling

- Reminder: all *scaling* should be done on training data and applied to test data to avoid leakage.
- For numerical reasons, it may help to *center* predictors.

a. Divide by *standard deviation* (after centering; z-score)
   - This puts all features in same units: standard deviations

$$x' = \frac{x - \hat{\mu}_x}{\hat{\sigma}_x} = \frac{x}{\hat{\sigma}_x} - \frac{\hat{\mu}_x}{\hat{\sigma}_x}$$

b. Range or max-min scaling:

$$x' = \frac{x - \min}{\max - \min}$$

   - Can be extremely influenced by outliers

c. Rank Scaling. Replace original values by their rank or sample quantiles

$$x' = \frac{\text{rank of } x}{n} = \frac{\text{\# obs } \leq x}{n}$$

   - Trees use this implicitly for splitting
   - Robust against outliers (but changes relationship between feature and outcome)
   - Can take further step to remap quantiles to make any distribution (inverse CDF)

3 Feature Selection

Note

- *Feature Selection*: only use a subset of available/collected predictors
  - E.g., best subsets
- *Dimension Reduction*: reduce the number of predictors/parameters used by a model
  - E.g., principal component regression (PCR)

See Feature Engineering: Selection for more details and ideas.

Goals:

1. Cost and time savings: Collecting predictors can be expensive
2. Reduce model variance: removing predictors lowers model variance
3. Interpretation: easier to understand model with fewer relevant predictors

Three main approaches:

1. Intrinsic
2. Wrappers
3. Filters

3.1 Intrinsic: Feature Selectors

Intrinsic feature selection methods are build into the model/algorithm. Examples include trees (may never split on a feature) and lasso (coefficients may be set to 0).

3.2 Wrappers: Feature Selectors

Wrapper methods for feature selection attempt to find the best subset of features for a particular model. They can *sometimes* perform better (e.g., Boruta) than intrinsic methods, but they will involve extra computation.

Examples include fully enumerated *best subsets*, *stepwise*, and evolutionary optimization algorithms like *genetic algorithms*. Can consider this a binary integer programming optimization. An example of *False Selection Rate (FSR)* feature selection designed for random forest is called **Boruta**.

Boruta for trees

- Introduce additional shuffled features (null features)
- Calculate importance scores for all features (real and null)
- Record the "hits": all original features with importance scores greater than *max importance from all null features* (these features are deemed important)
- Repeat the process M times (100 by default; or consider sequential approaches)
- Determine which predictors have significantly more "hits" than expected under the null of not-important.

3.3 Filters: Feature Selectors

Filter methods are the quickest, but not usually the best. Basically, filter methods do feature selection first before any modeling is done. For example, run p simple linear regression models (one for each predictor) and keep the features with significant coefficients for further modeling.

Although they are sometimes claimed to be model-free, most (all?) filter methods do (implicitly) have a model they are using to decide on the relevant predictors. There is no guarantee that features selected by the filter are appropriate for the final model.

Also, need to be very careful with data leakage; using a supervised filter method **before** resampling will give false sense of model performance.

But *unsupervised* filtering (e.g., removing stop words, almost zero-variance predictors, removing duplicate or highly correlated predictors) can prevent inflated variance.

4 Dimension Reduction

Note

- *Feature Selection*: only use a subset of available/collected predictors
  - E.g., best subsets
- *Dimension Reduction*: reduce the number of predictors/parameters *used by the model*
  - E.g., principal component regression (PCR)
  - The predictors used by the model may not be the same ones that were collected.

Basically, dimension reduction methods are based on transforming the raw data (e.g., PCA) and then using a subset of the transformed predictions.

Principal Component Regression (PCR) and Partial Least Squares (PLS) are classic examples of dimension reduction.

- These don't actually remove predictors since using linear combination. So you can gain on reduced model variance, but don't get easier interpretation and still need to collect all input predictors.

4.1 Linear Regression (OLS)

The standard generic form for a linear regression model is

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots, + \beta_p X_p + \epsilon$$

- Y is the response or dependent variable
- $X_1, X_2, \ldots, X_p$ are called the p explanatory, independent, or predictor variables
- the greek letter ϵ (epsilon) is the random error variable

Linear Model Diagram

4.2 Estimation

- The weights/coefficients (β) are the *model parameters*

- OLS uses the weights/coefficients that minimize the RSS loss function over the training data

$$\begin{aligned}\hat{\beta} &= \arg\min_{\beta} \ \text{RSS}(\beta) \qquad \text{Note: } \beta \text{ is a } \textit{vector} \\ &= \arg\min_{\beta} \ \sum_{i=1}^{n}(y_i - f(\mathbf{x}_i;\beta))^2 \\ &= \arg\min_{\beta} \ \sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_{i1} - \beta_2 x_{i2} + \ldots + \beta_p x_{ip})^2\end{aligned}$$

OLS equivalently minimizes the MSE since $\text{MSE} = \text{RSS}/n$.

4.2.1 Matrix notation

$$f(\mathbf{x};\beta) = \mathbf{x}^\mathsf{T}\beta$$

$$Y = \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_n \end{bmatrix} \qquad X = \begin{bmatrix} 1 & X_{11} & X_{12} & X_{13} & \ldots & X_{1p} \\ 1 & X_{21} & X_{22} & X_{23} & \ldots & X_{2p} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & X_{n1} & X_{n2} & X_{n3} & \ldots & X_{np} \end{bmatrix} \qquad \beta = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}$$

$$\text{RSS}(\beta) = (\text{Y} - \text{X}\beta)^\mathsf{T}(\text{Y} - \text{X}\beta)$$

$$\begin{aligned}\frac{\partial \text{RSS}(\beta)}{\partial \beta} &= 2X^\mathsf{T}(Y - X\beta) \\ &\implies X^\mathsf{T}Y = X^\mathsf{T}X\beta \\ &\implies \boxed{\hat{\beta} = (X^\mathsf{T}X)^{-1}X^\mathsf{T}Y}\end{aligned}$$

4.3 Some Problems with least squares estimates

There are a few problems with using least squares estimation (OLS) to estimate the regression parameters (coefficients)

- *Prediction Accuracy*
  - the least squares estimates in high dimensional data may have low bias but can suffer from large variance.
  - Prediction accuracy can sometimes be improved by shrinking or setting some coefficients to zero.
  - By doing so we sacrifice a little bit of bias to reduce the variance of the predicted values, and hence may improve the overall prediction accuracy.
  - Some predictors may not have any predictive value and only increase noise

- *Interpretation*: With a large number of predictors, we often would like to determine a smaller subset that exhibit the strongest effects. In order to get the "big picture", we are willing to sacrifice some of the small details
  - When $p > n$ least squares won't work at all

4.4 Improving Least squares

We will examine 3 standard approaches to improve on least squares estimates

1. Subset Selection
   - Only use a subset of predictors, but estimate with OLS

- Examples: *best subsets*, *forward step-wise*

2. Shrinkage/Penalized/Regularized Regression
   - Instead of an "all or nothing" approach, shrinkage methods force the coefficients closer toward 0.
   - Examples: *ridge*, *lasso*, *elastic net*
3. Dimension Reduction with Derived Inputs
   - Use a subset of linearly transformed predictors
   - Examples: *PCA*, *PLS*

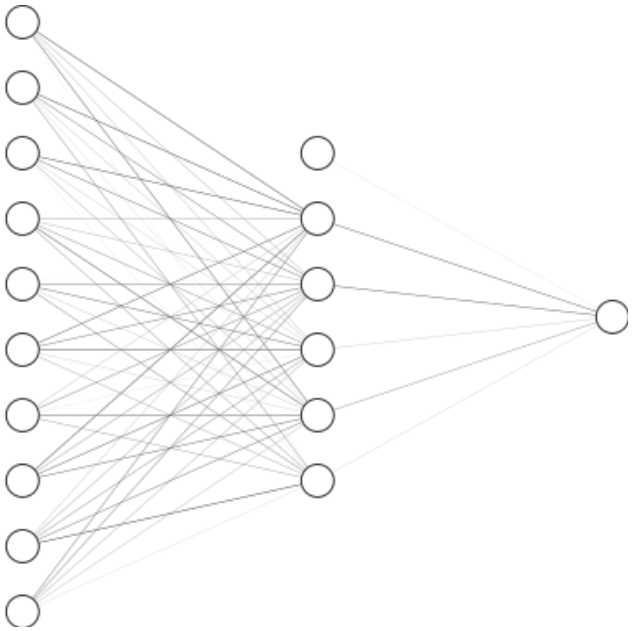
All three methods introduce some additional bias in order to reduce variance and *hopefully* improve prediction.

4.5 Derived Linear Features

Instead of using the raw features as predictors, it can sometimes be helpful to use derived features (e.g., new features as transformations of the raw features).

- X is the $(n \times p)$ raw predictor matrix
  - p predictors
- Z is the $(n \times r)$ derived predictor matrix
  - r predictors
  - r could be less than (*dimension reduction*) , equal to, or greater than p (*feature expansion*)
- We saw *feature expansion* (i.e., basis expansion) when we used splines and polynomials to allow non-linear relationship between outcome and single predictor
- Today's material is more focused on *dimension reduction* ($r < p$) as a way to introduce some bias to reduce variance
  - Just like we did with penalized regression (e.g., ridge, lasso, elasticnet)

4.5.1 Linear Transformations



- Let $Z = XA$ be the $(n \times r)$ transformed model matrix
  - X is the $(n \times p)$ original features

- A is the $(p \times r)$ linear transformation matrix
- A_m is the mth column of A
- a_{jm} is the (j, m) element of A

$$
\begin{aligned}
Z &= XA \\
Z_m &= XA_m \\
&= \sum_{j=1}^{p} X_j a_{jm} \\
Z_{im} &= \sum_{j=1}^{p} X_{ij} a_{jm}
\end{aligned}
$$

4.5.2 OLS with derived feature model

- Once we have the new feature matrix Z, we can estimate parameters like usual. For example, with OLS:

$$\boxed{\hat{\theta} = (Z^\mathsf{T} Z)^{-1} Z^\mathsf{T} Y}$$

This gives predictions for raw input x:

$$\hat{y}(x) = \hat{\theta}_0 + \sum_{m=1}^{r} Z_m(x) \hat{\theta}_m$$

Plugging in $Z_m(x) = \sum_{j=1}^{p} x_j a_{jm}$:

Estimated Derived Beta Coefficients

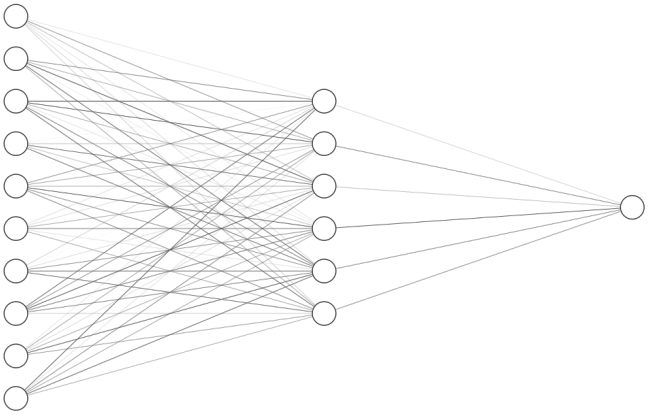
4.5.3 Dimension Reduction vs. Feature Selection

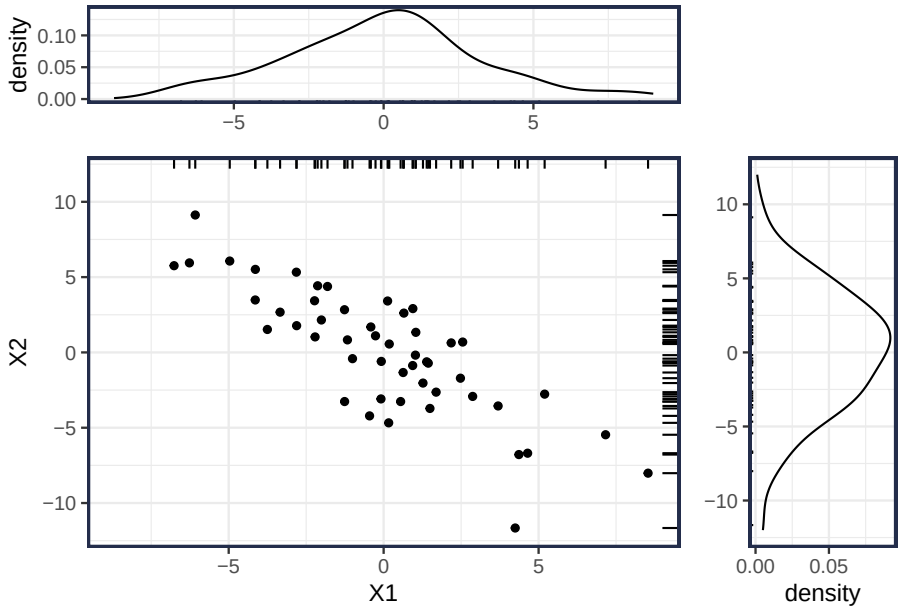
If $r < p$, then fewer model parameters need to be estimated. This is called *dimension reduction* since we have less parameters to estimate.

- Hence, edf is decreased (lower variance, higher bias)

However, because we still use all original features we haven't actually done *feature selection*, so all raw features must still be collected.

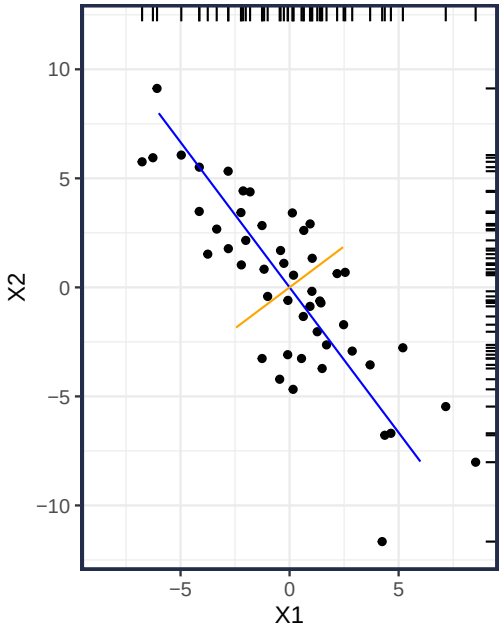
5 Principal Component Regression (PCR)

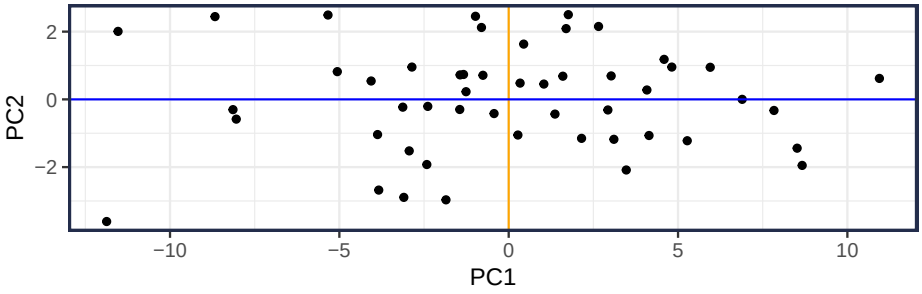




Variance-Covariance Matrix of (centered) data:

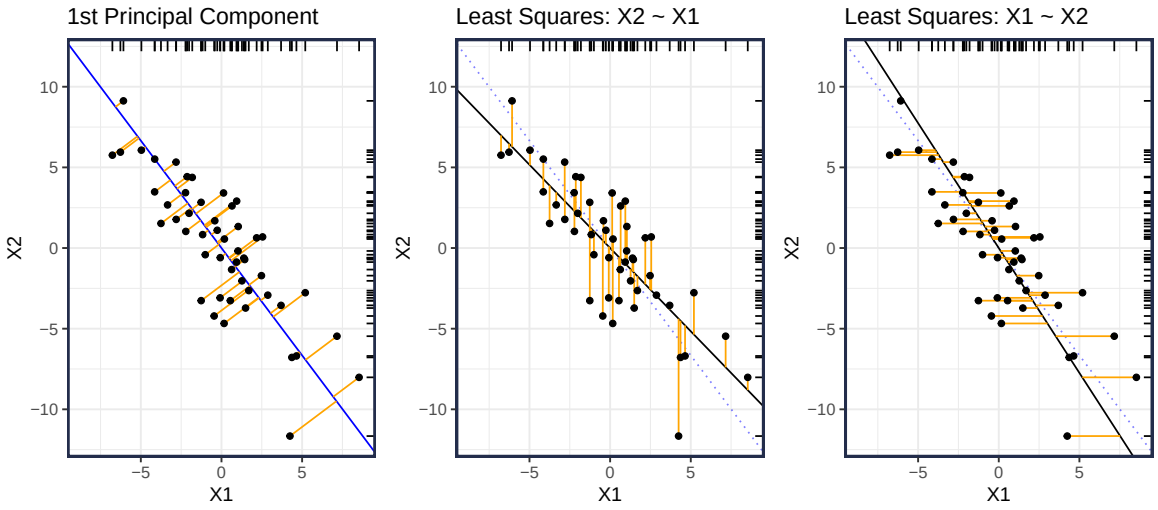
	X1	X2
X1	10.50	-10.84
X2	-10.84	16.80





Variance-Covariance Matrix of Principal Component projections:

	PC1	PC2
PC1	24.94	0.00
PC2	0.00	2.36



5.1 Eigen Decomposition (Spectral Analysis)

5.2 Principal Component Analysis (PCA)

5.3 Dimension Reduction with PCR

6 Singular Value Decomposition (SVD)

7 PCA with SVD

8 Ridge Regression with SVD

9 Comparison