Statistical Learning with Sparsity

Matrix Completion

2019 Wald Lecture II

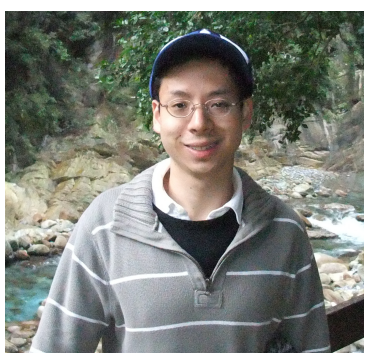
Trevor Hastie
Department of Statistics
Department of Biomedical Data Science
Stanford University



Outline of Talk

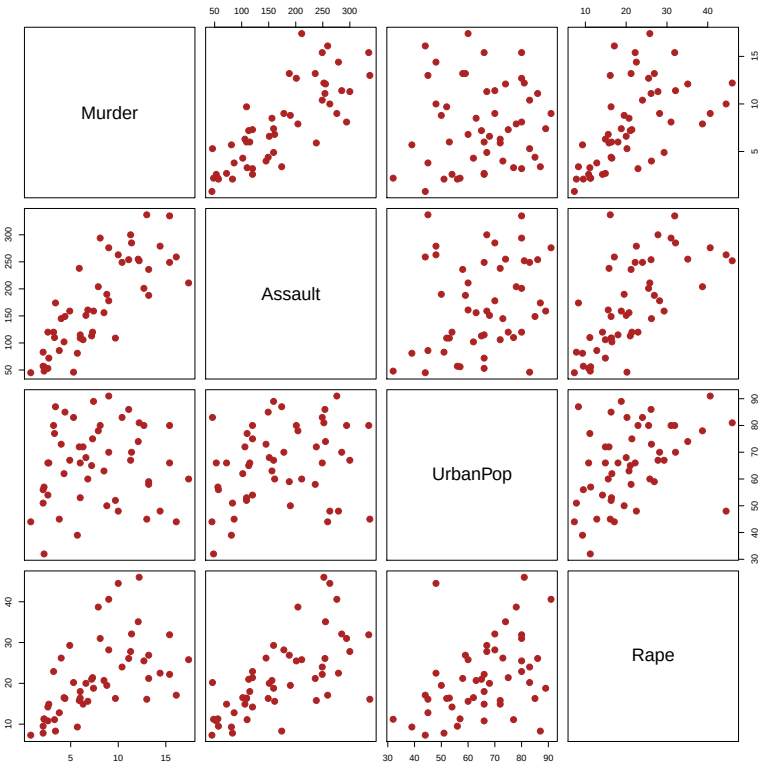
- Convex matrix completion, collaborative filtering
  (Mazumder, H, Tibshirani 2010 JMLR)
- Recent algorithmic advances and large-scale SVD
  (H, Mazumder, Lee, Zadeh 2015 JMLR)
- Longitudinal Data Analysis using Matrix Completion
  (Kidzinski, H, 2018 arXiv)

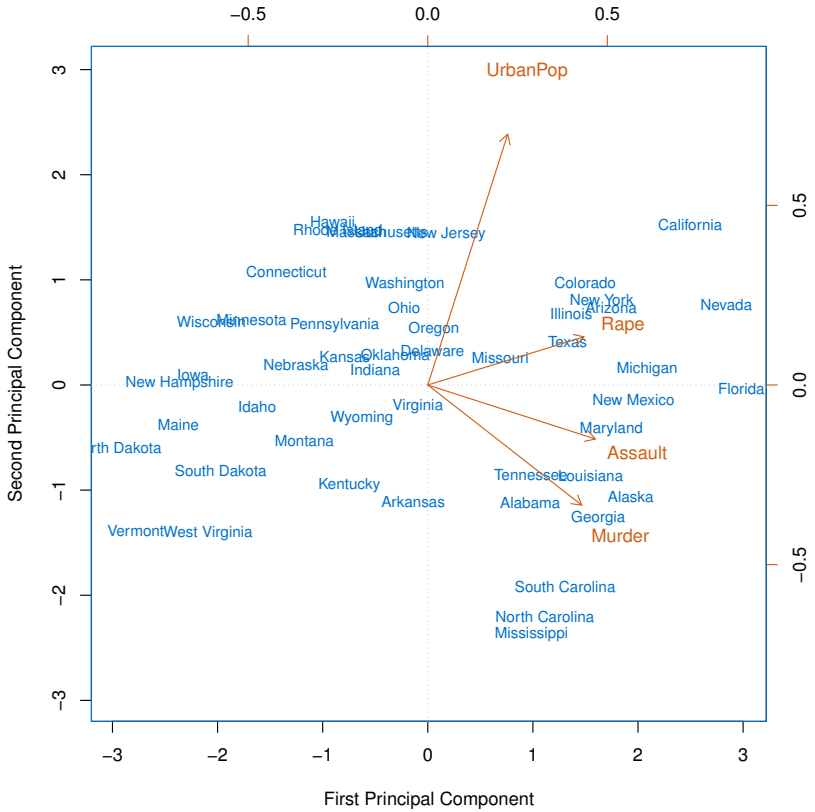






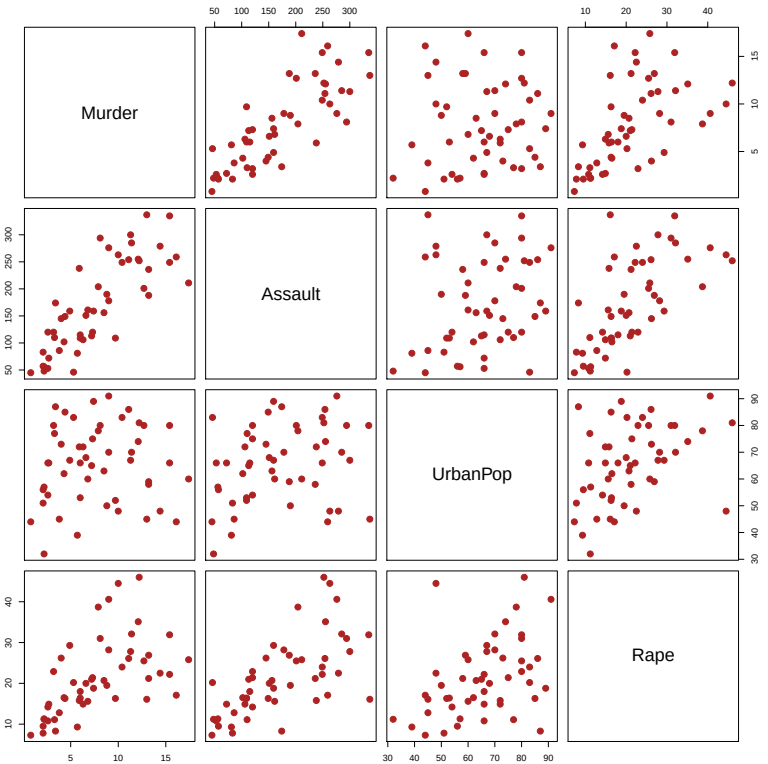
Principal Components

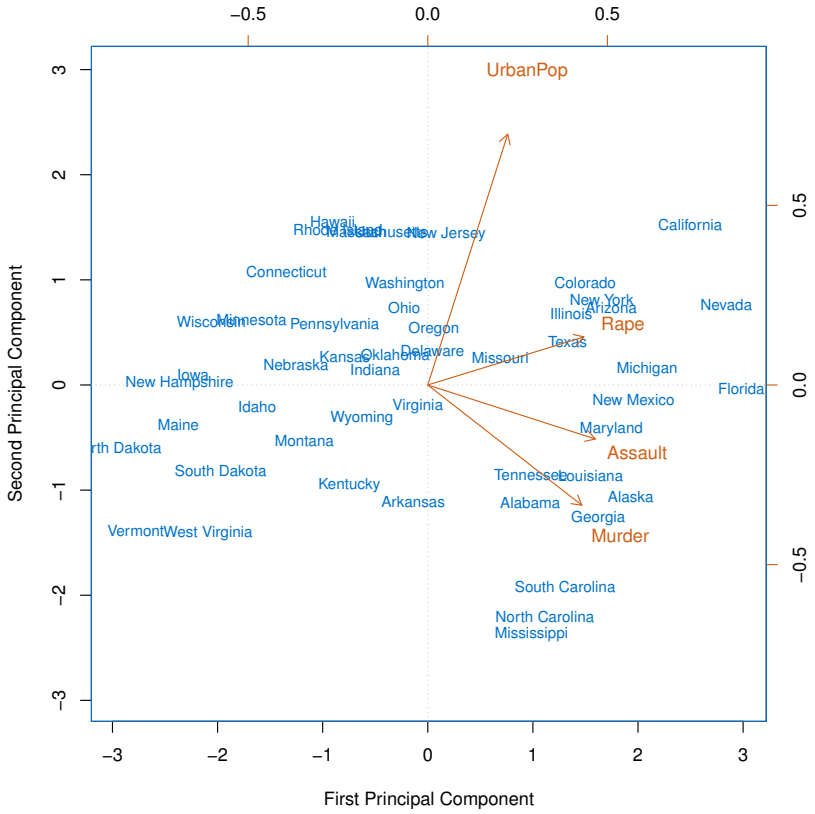




- Data reduction with correlated variables.
- Biplots and low-dimensional representations.

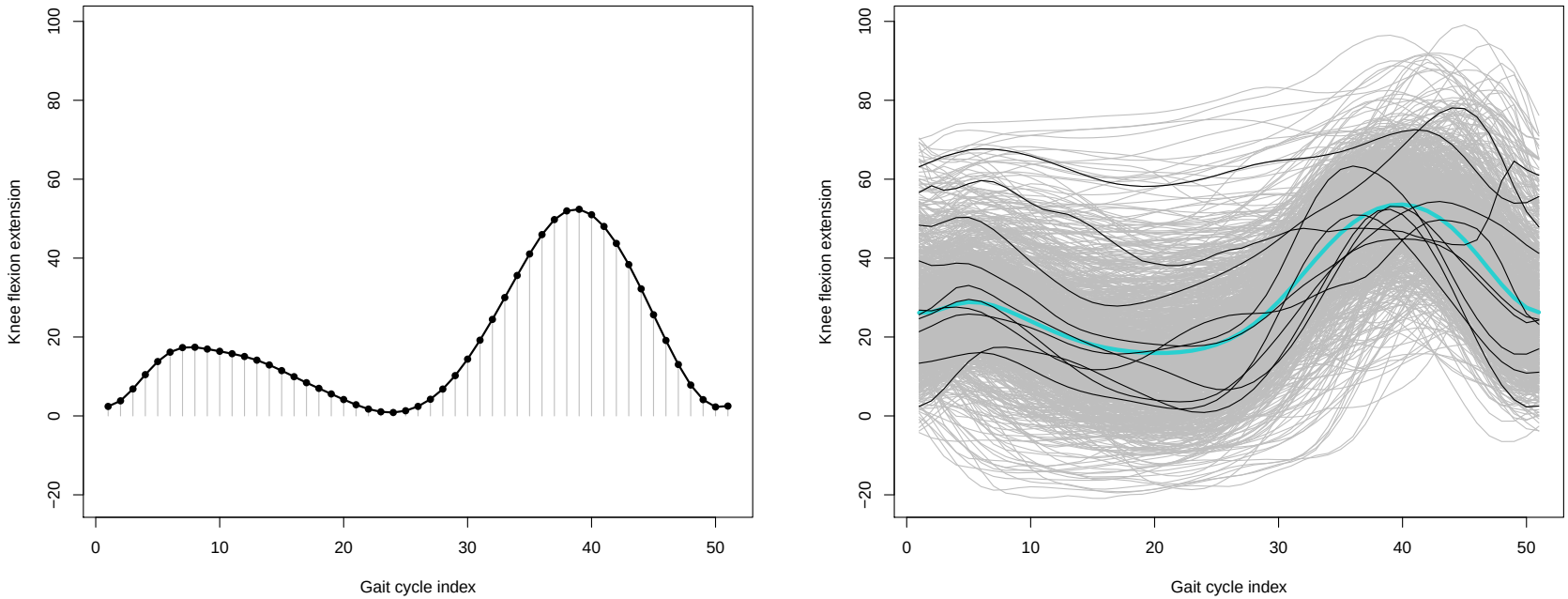
Principal Components





- Data reduction with correlated variables.
- Biplots and low-dimensional representations.
- *What if some entries in data matrix are missing?*

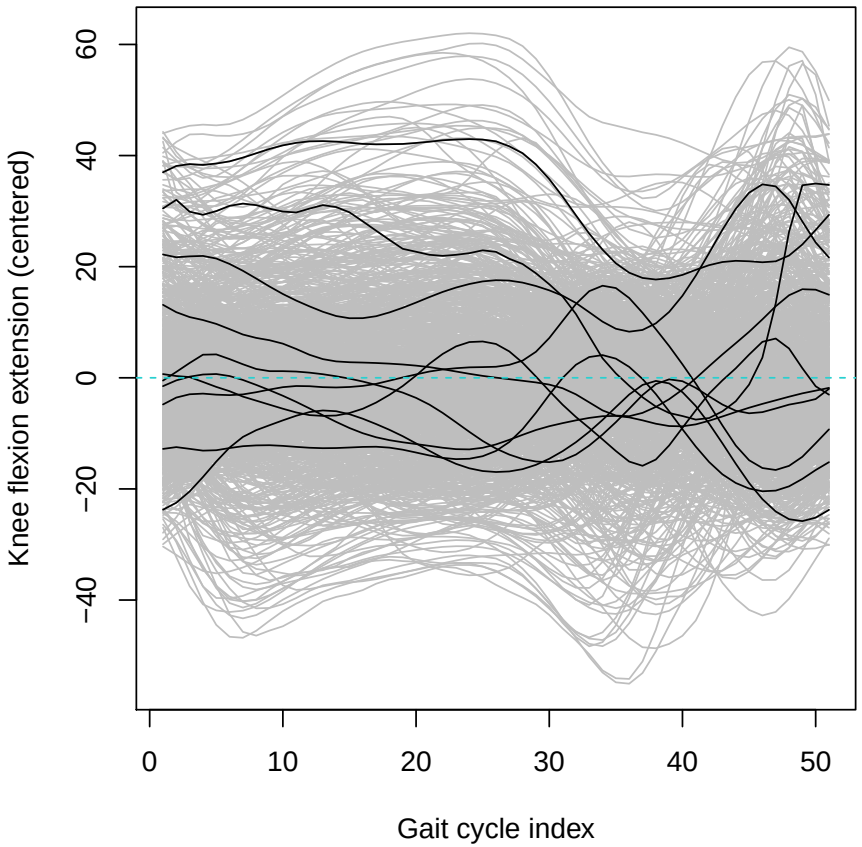
Functional Principal Components

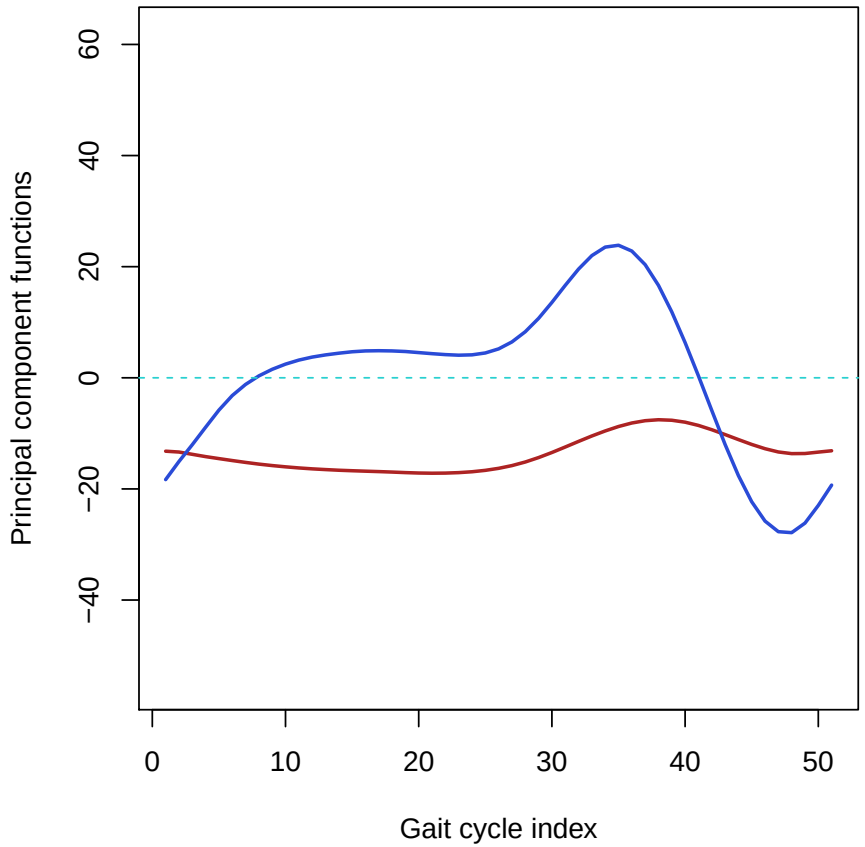


Knee Flexion Extension — Cerebral Palsy Patients

- Turquoise curve is average
- High between-subject variation

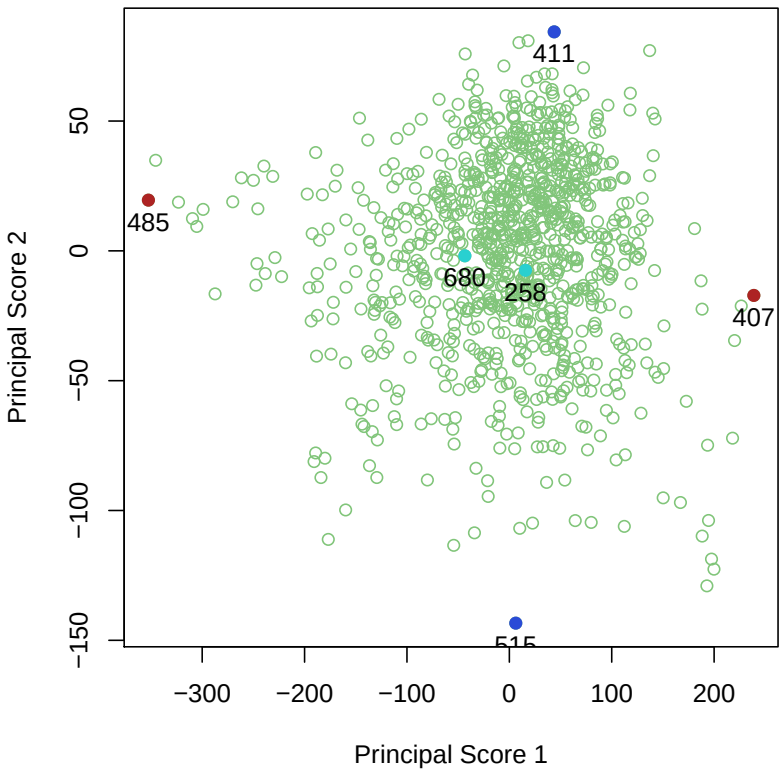
First two functional principal components

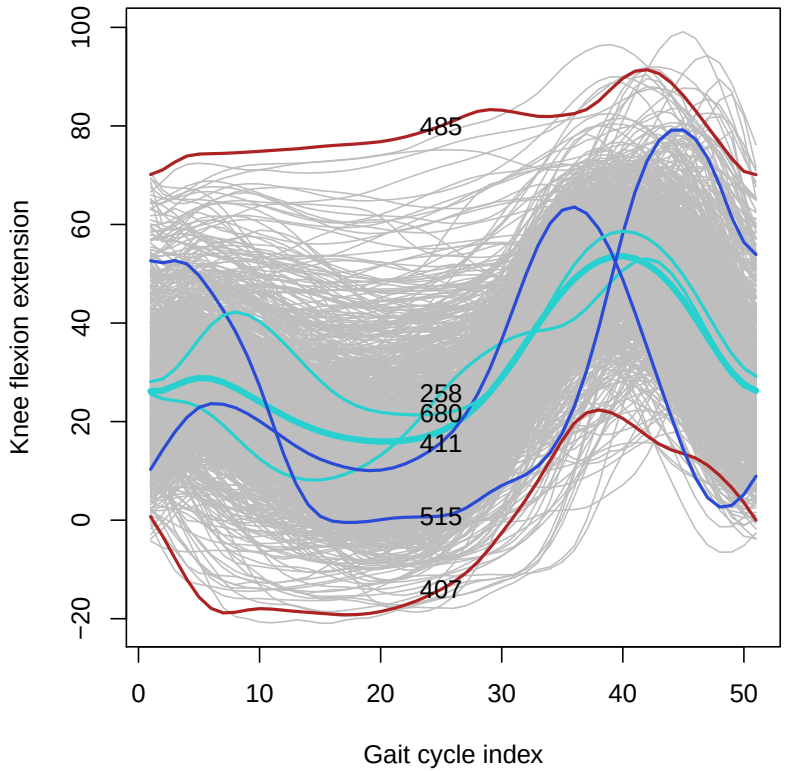




$$x_i \approx \bar{x} + s_{1i}v_1 + s_{2i}v_2$$

Functional principal component scores

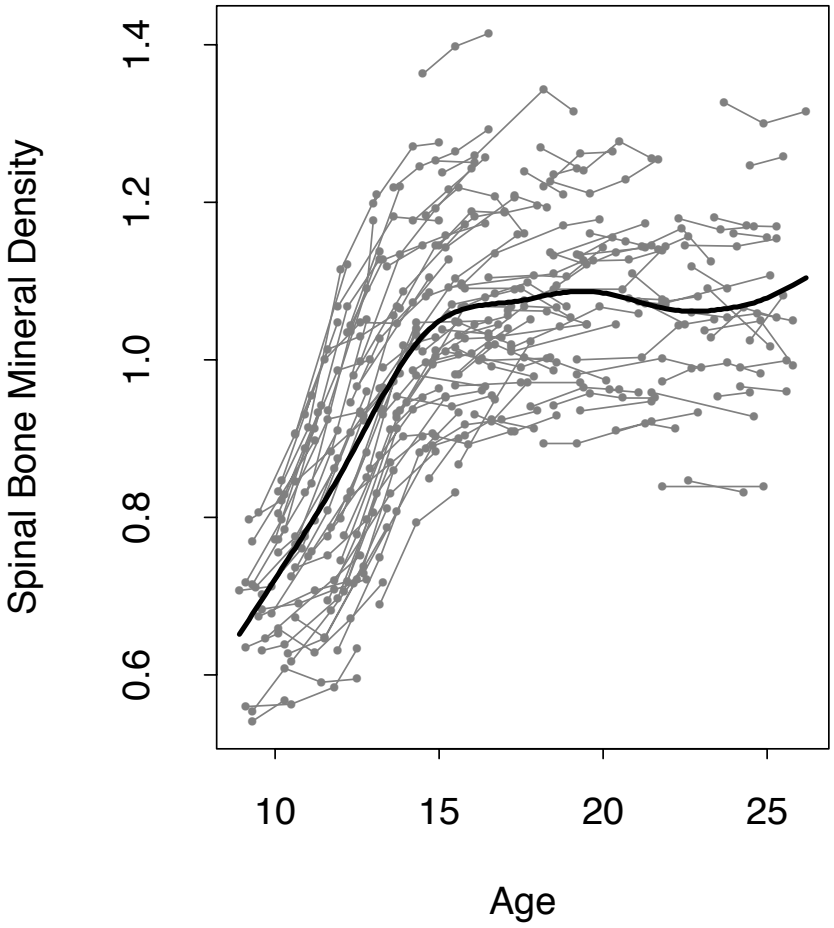


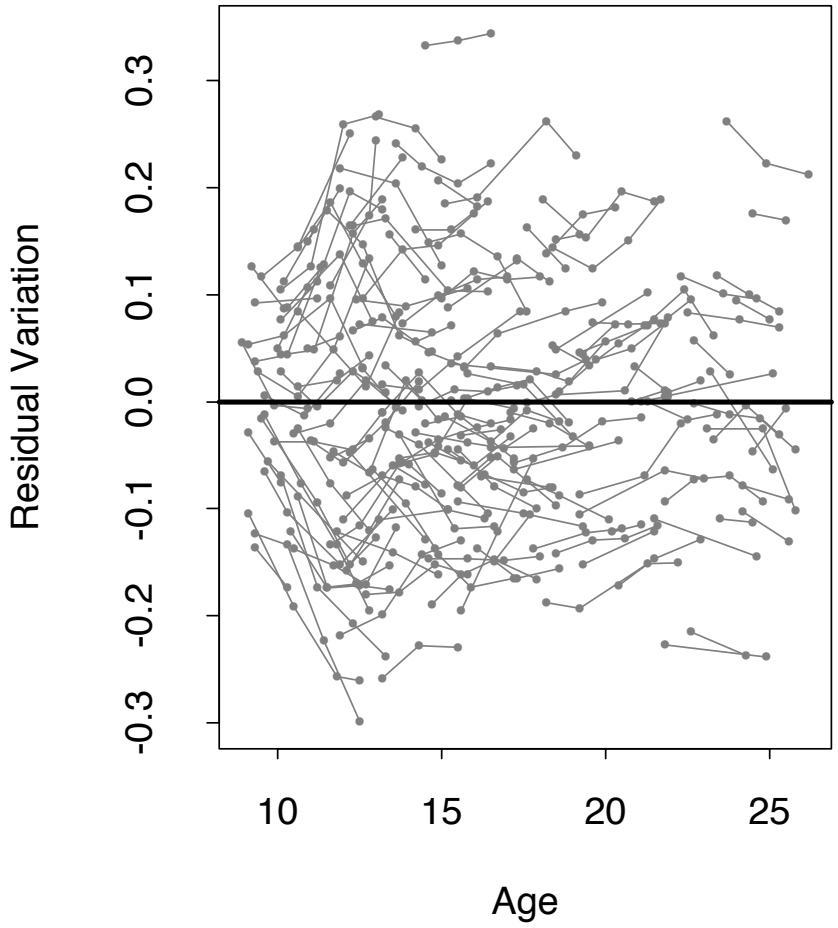


$$x_i \approx \bar{x} + s_{1i}v_1 + s_{2i}v_2$$

Scores are the (s_{1i}, s_{2i}) for each sampled curve x_i.

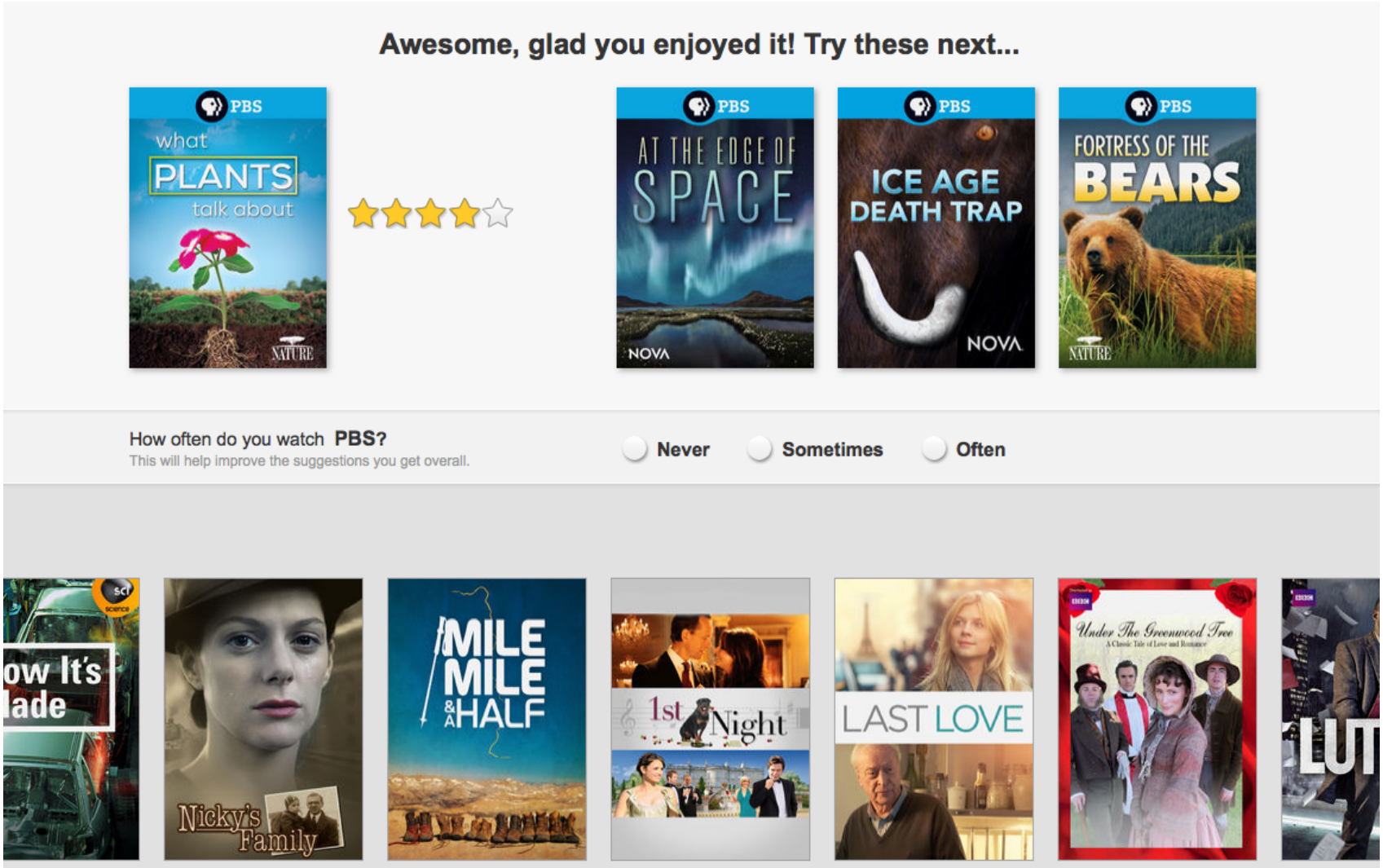
Sparse Functional Data



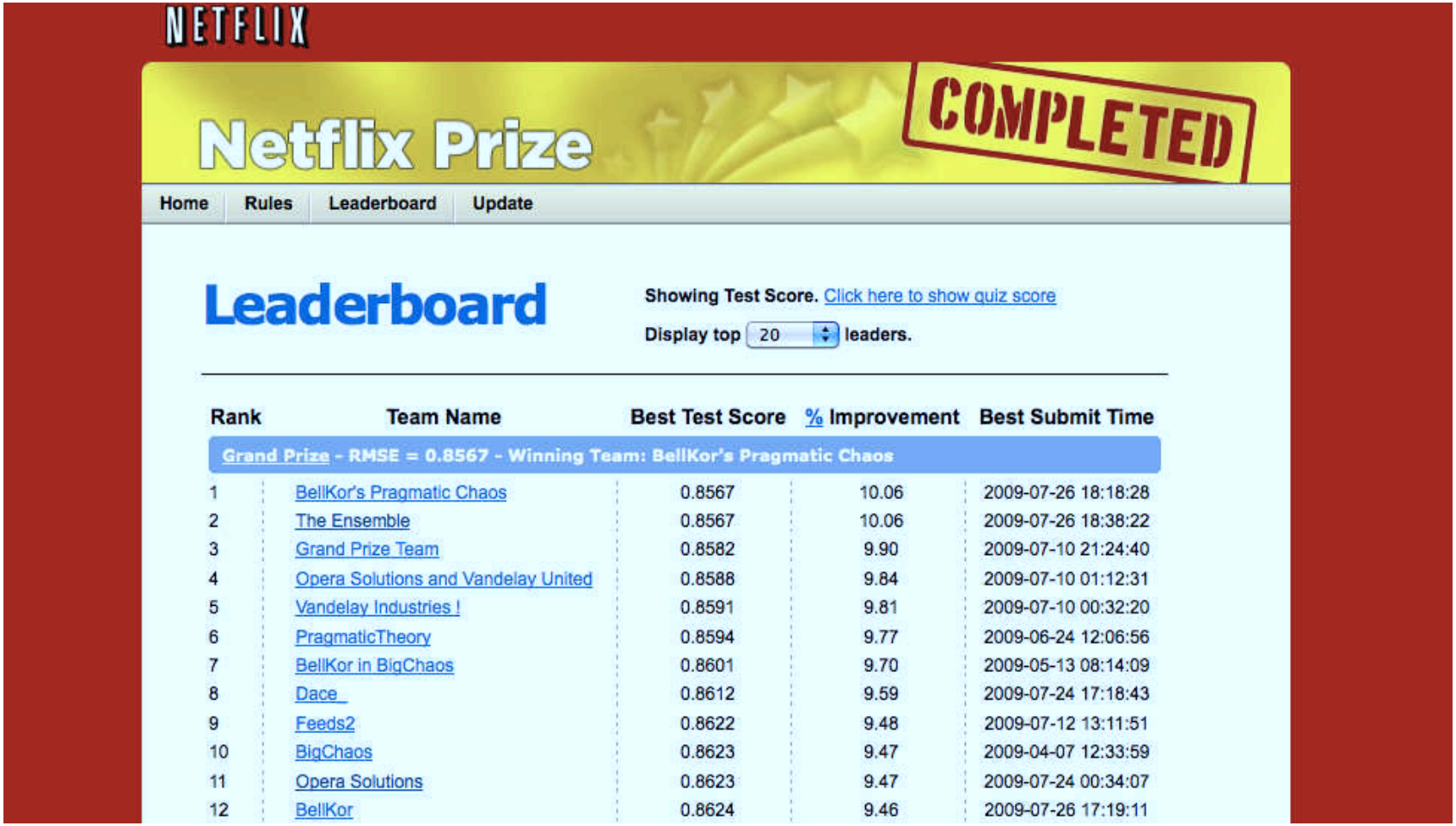


- Only a fragment of a curve for each girl.
- *Mostly missing data!*

Recommender Systems



The Netflix Prize — 2006–2009



NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

41K teams participated!

Winner *BellKor's Pragmatic Chaos*, essentially tied with *The Ensemble*

The Netflix Prize — 2006–2009

NETFLIX

Netflix Prize

COMPLETED

Home | Rules | Leaderboard | Update

Leaderboard

Showing Test Score. Click here to show quiz score

Display top 20 leaders.

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07
12	BellKor	0.8624	9.46	2009-07-26 17:19:11

41K teams participated!

Winner *BellKor's Pragmatic Chaos*, essentially tied with *The Ensemble* $\supset$ *our Lester Mackey* $\rightarrow$



Netflix Data Set

	Dirty Dancing	Meet the Parents	Top Gun	The Sixth Sense	Catch Me If You Can	The Royal Tenenbaums	Con Air	Big Fish	The Matrix	A Few Good Men
Customer 1	·	·	·	·	4	·	·	·	·	·
Customer 2	·	·	3	·	·	·	3	·	·	3
Customer 3	·	2	·	4	·	·	·	·	2	·
Customer 4	3	·	·	·	·	·	·	·	·	·
Customer 5	5	5	·	·	4	·	·	·	·	·
Customer 6	·	·	·	·	·	2	4	·	·	·
Customer 7	·	·	5	·	·	·	·	3	·	·
Customer 8	·	·	·	·	·	2	·	·	·	3
Customer 9	3	·	·	·	5	·	·	5	·	·
Customer 10	·	·	·	·	·	·	·	·	·	·

Netflix Data Set

	Dirty Dancing	Meet the Parents	Top Gun	The Sixth Sense	Catch Me If You Can	The Royal Tenenbaums	Con Air	Big Fish	The Matrix	A Few Good Men
Customer 1	·	·	·	·	4	·	·	·	·	·
Customer 2	·	·	3	·	·	·	3	·	·	3
Customer 3	·	2	·	4	·	·	·	·	2	·
Customer 4	3	·	·	·	·	·	·	·	·	·
Customer 5	5	5	·	·	4	·	·	·	·	·
Customer 6	·	·	·	·	·	2	4	·	·	·
Customer 7	·	·	5	·	·	·	·	3	·	·
Customer 8	·	·	·	·	·	2	·	·	·	3
Customer 9	3	·	·	·	5	·	·	5	·	·
Customer 10	·	·	·	·	·	·	·	·	·	·

$480K$ Customers, $18K$ Movies, $100M$ ratings (1-5)

$8B$ cells (99% NA)

The Netflix Challenge

- Goal: $1M prize for 10% reduction in RMSE over Netflix Cinematch algorithm.
- Teams could submit predictions once a day, which were scored on a left-out evaluation set.
- BellKor's Pragmatic Chaos declared winners on 9/21/2009
- Used ensemble of models, an important ingredient being low-rank SVD
- SVD $\equiv$ PCA, here with many missing values

SVD, PCA and Matrix Approximation

$$\min_{Z} \|X - Z\|_F \quad \text{s.t. } \operatorname{rank}(Z) \leq r$$

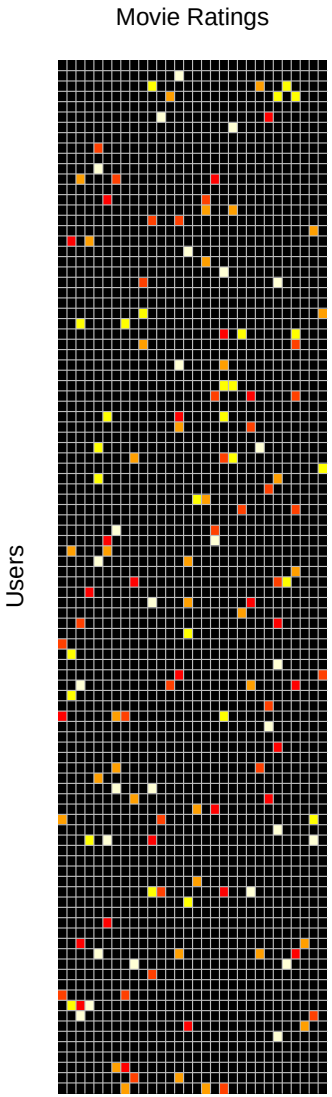
where X is centered data matrix.

- Solution given by SVD of $X = UDV^\top$:

$$\begin{aligned} \widehat{Z} &= U_r D_r V_r^\top \\ &= S_r V_r^\top \end{aligned}$$

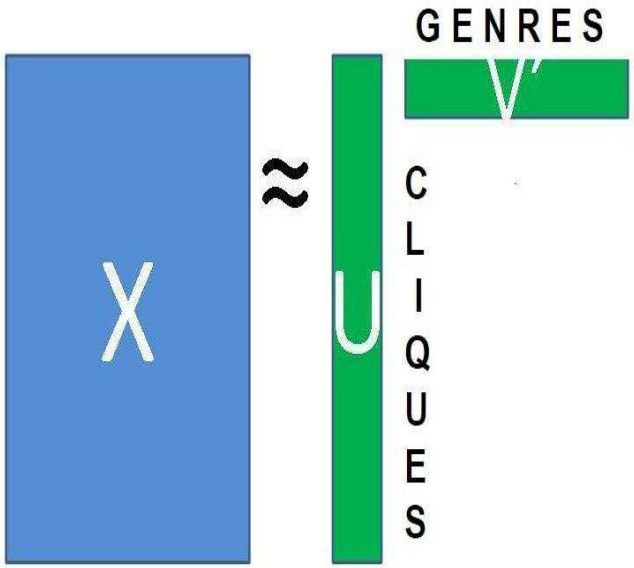
- $S_r = U_r D_r$ is matrix of largest r principal components, and the columns of V_r are the corresponding r PC directions.
- With missing data, we solve the PCA problem via the objective above, and get predictions for the missing entries.

Matrix Completion/ Collaborative Filtering



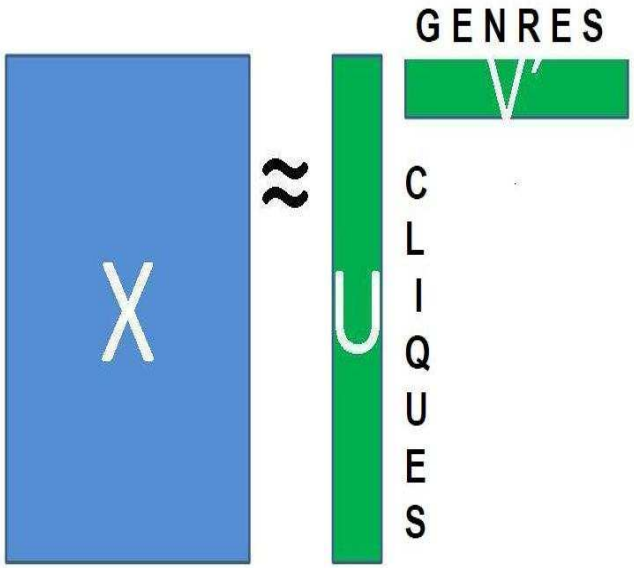
- Large matrices
  # rows | #columns $\approx 10^5, 10^6$
- Very under-determined
  (often only 1–2% observed)
- Exploit matrix structure, row | column interactions
- Task: "fill-in" missing entries
- Applications: recommender systems, image-processing, functional PCA, imputation of NAs for genomic data, rank estimation for SVD, and many more.

Model Assumption : Low Rank + Noise



- Under-determined — assume low-rank
- Why low rank?

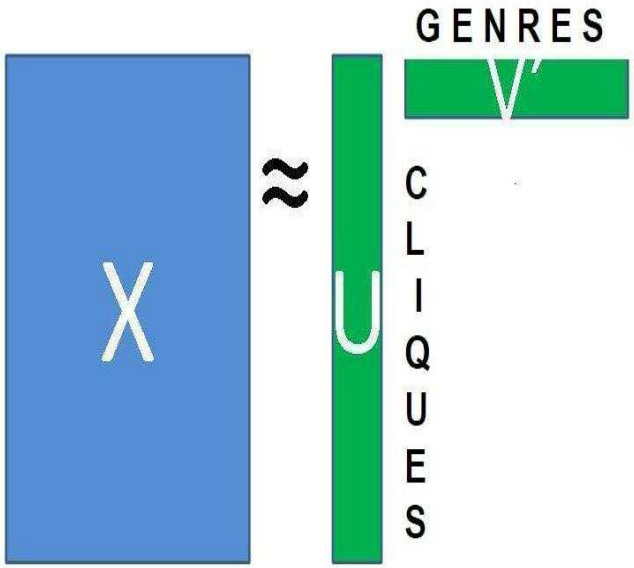
Model Assumption : Low Rank + Noise



- Under-determined — assume low-rank
- Why low rank?

  Interpretation — User & item factors induce collaboration

Model Assumption : Low Rank + Noise

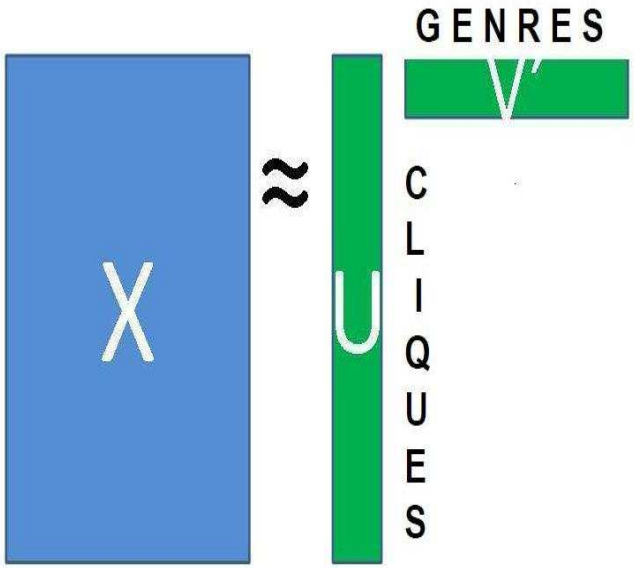


- Under-determined — assume low-rank
- Why low rank?

  Interpretation — User & item factors induce collaboration

  Empirical — Netflix successes

Model Assumption : Low Rank + Noise



- Under-determined — assume low-rank
- Why low rank?

  Interpretation — User & item factors induce collaboration

  Empirical — Netflix successes

  Theoretical — "Reconstruction" possible under low-rank & regularity conditions

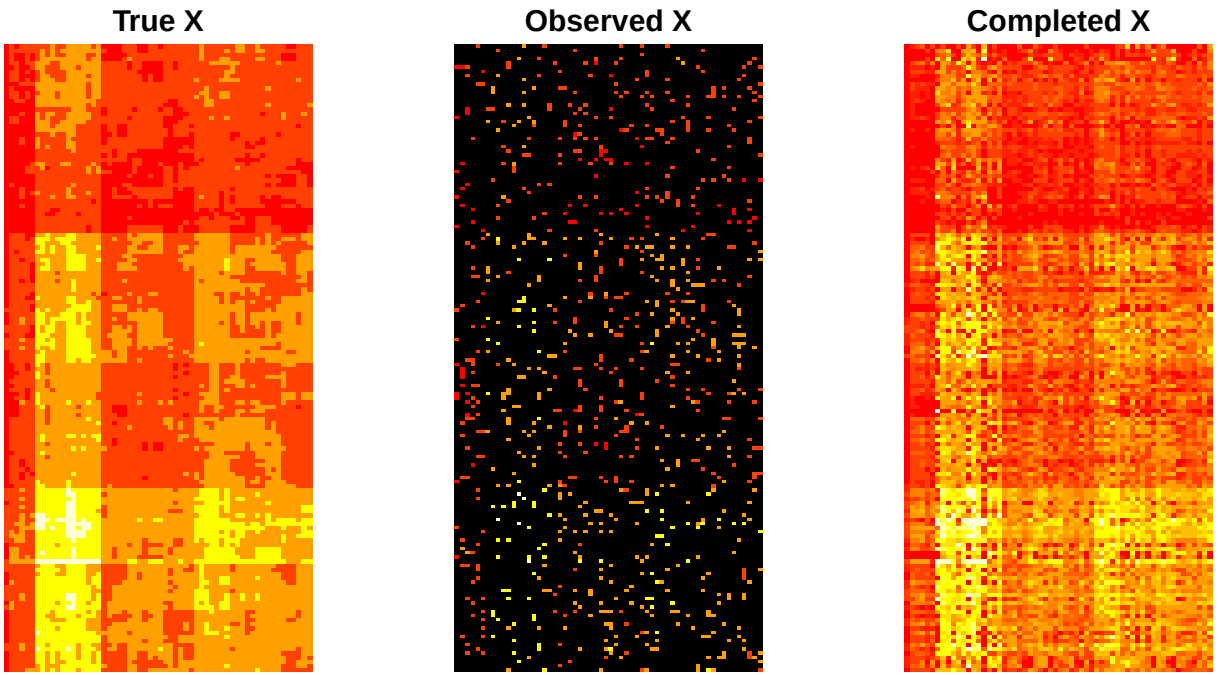
  Srebro et al (2005); Candes and Recht (2008); Candes and Tao (2009); Keshavan et. al. (2009); Negahban and Wainwright (2012)

Optimization problem

Find $Z_{n\times m}$ of (small) rank r such that training error is small.

$$\underset{Z}{\text{minimize}} \sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 \quad \text{subject to } \text{rank}(Z) \leq r$$

Impute missing X_{ij} with $\widehat{Z}_{ij}$



m=150, n=60, r=5, 90% missing — hardImpute algorithm.

hardImpute

Very simple idea for fitting a rank r SVD to a matrix X in the presence of NAs.

hardImpute

0. Complete X by filling in NAs in some way (eg 0s).

Now repeat steps 2 and 3 till convergence.

1. Compute rank-r SVD approximation to completed X.
2. Complete X using current rank-r fit.

hardImpute

Very simple idea for fitting a rank r SVD to a matrix X in the presence of NAs.

hardImpute

0. Complete X by filling in NAs in some way (eg 0s).

Now repeat steps 2 and 3 till convergence.

1. Compute rank-r SVD approximation to completed X.
2. Complete X using current rank-r fit.

Works surprisingly well!

hardImpute

Very simple idea for fitting a rank r SVD to a matrix X in the presence of NAs.

hardImpute

0. Complete X by filling in NAs in some way (eg 0s).

Now repeat steps 2 and 3 till convergence.

1. Compute rank-r SVD approximation to completed X.
2. Complete X using current rank-r fit.

Works surprisingly well!
But no guarantees since the problem is non-convex

hardImpute

Very simple idea for fitting a rank r SVD to a matrix X in the presence of NAs.

hardImpute

0. Complete X by filling in NAs in some way (eg 0s).

Now repeat steps 2 and 3 till convergence.

1. Compute rank-r SVD approximation to completed X.
2. Complete X using current rank-r fit.

Works surprisingly well!
But no guarantees since the problem is non-convex
Has been reinvented many times!

Convex Approach: Nuclear Norm Relaxation

- The rank(Z) constraint makes the problem non-convex — combinatorially hard (although good algorithms such as hardImpute exist).
- $\|Z\|_* = \sum_j \lambda_j(Z)$ — sum of singular values of Z — is convex in Z. Called the "nuclear norm" of Z.
- $\|Z\|_*$ tightest convex relaxation of rank(Z) (Fazel, Boyd, 2002)

We solve instead

$$\underset{Z}{\text{minimize}} \sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 \quad \text{subject to } \|Z\|_* \leq \tau$$

which is convex in Z.

Convex Approach: Nuclear Norm Relaxation

- The rank(Z) constraint makes the problem non-convex — combinatorially hard (although good algorithms such as hardImpute exist).
- $\|Z\|_* = \sum_j \lambda_j(Z)$ — sum of singular values of Z — is convex in Z. Called the "nuclear norm" of Z.
- $\|Z\|_*$ tightest convex relaxation of rank(Z) (Fazel, Boyd, 2002)

We solve instead

$$\underset{Z}{\text{minimize}} \sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 \quad \text{subject to } \|Z\|_* \leq \tau$$

which is convex in Z.

Convex Approach: Nuclear Norm Relaxation

- The rank(Z) constraint makes the problem non-convex — combinatorially hard (although good algorithms such as hardImpute exist).
- $\|Z\|_* = \sum_j \lambda_j(Z)$ — sum of singular values of Z — is convex in Z. Called the "nuclear norm" of Z.
- $\|Z\|_*$ tightest convex relaxation of rank(Z) (Fazel, Boyd, 2002)

We solve instead

$$\underset{Z}{\text{minimize}} \sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 \quad \text{subject to } \|Z\|_* \leq \tau$$

which is convex in Z.

Convex Approach: Nuclear Norm Relaxation

- The rank(Z) constraint makes the problem non-convex — combinatorially hard (although good algorithms such as hardImpute exist).
- $\|Z\|_* = \sum_j \lambda_j(Z)$ — sum of singular values of Z — is convex in Z. Called the "nuclear norm" of Z.
- $\|Z\|_*$ tightest convex relaxation of rank(Z) (Fazel, Boyd, 2002)

We solve instead

$$\underset{Z}{\text{minimize}} \sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 \quad \text{subject to } \|Z\|_* \leq \tau$$

which is convex in Z.

Notation

Let Ω be a mask for missing entries — a binary matrix the same shape as X:

$$\Omega_{i,j} = \begin{cases} 1 & \text{if } (i,j) \text{ is observed} \\ 0 & \text{if } (i,j) \text{ is missing} \end{cases}$$

Criterion rewritten as:

$$\sum_{\text{Observed}(i,j)} (X_{ij} - Z_{ij})^2 = \|\Omega \circ (X - Z)\|_F^2$$

with the convention that $0 \times \text{NA} = 0$.
The Hadamard product operates elementwise.

Soft SVD — Prox operator for Nuclear Norm

Let (fully observed) $X_{n\times m}$ have SVD

$$X = U \cdot \text{diag}[d_1, \ldots, d_m] \cdot V^\top$$

Consider the convex optimization problem

$$\underset{Z}{\text{minimize}} \quad \frac{1}{2}\|X - Z\|_F^2 + \lambda\|Z\|_*$$

Solution is *soft-thresholded SVD*

$$\mathbf{S}_\lambda(X) \;:=\; U \cdot \text{diag}[(d_1 - \lambda)_+, \ldots, (d_m - \lambda)_+] \cdot V^\top$$

Like *lasso* for SVD: singular values are shrunk to zero, with many set to zero.

Soft SVD — Prox operator for Nuclear Norm

Let (fully observed) $X_{n\times m}$ have SVD

$$X = U \cdot \text{diag}[d_1, \ldots, d_m] \cdot V^\top$$

Consider the convex optimization problem

$$\underset{Z}{\text{minimize}} \quad \frac{1}{2}\|X - Z\|_F^2 + \lambda\|Z\|_*$$

Solution is *soft-thresholded SVD*

$$\mathbf{S}_\lambda(X) \;:=\; U \cdot \text{diag}[(d_1 - \lambda)_+, \ldots, (d_m - \lambda)_+] \cdot V^\top$$

Like *lasso* for SVD: singular values are shrunk to zero, with many set to zero.
Smooth version of best-rank approximation.

Convex Optimization Problem

Back to missing data problem, in Lagrange form:

$$\underset{Z}{\text{minimize}} \ \frac{1}{2}\|\Omega \circ (X - Z)\|_F^2 + \lambda \|Z\|_*$$

- This is a semi-definite program (SDP), convex in Z.
- Complexity of existing off-the-shelf solvers:
  - interior-point methods: $O(n^4) \ldots O(n^6) \ldots$
  - (black box) first-order methods complexity: $O(n^3)$
- We solve using an iterative soft SVD (next slide), with cost per soft SVD $O[(m+n) \cdot r + |\Omega|]$ where r is rank of solution.

softImpute

Algorithm to solve the convex matrix completion problem

$$\underset{Z}{\text{minimize}} \ \frac{1}{2}\|\Omega \circ (X - Z)\|_F^2 + \lambda \|Z\|_*$$

softImpute

0. Initialize Z in some way (eg 0s or warm start).

Repeat till convergence.

1. Compute $Z^{\text{new}} \leftarrow \mathbf{S}_\lambda \left(\Omega \circ X + (1 - \Omega) \circ Z^{\text{old}}\right)$.

softImpute

Algorithm to solve the convex matrix completion problem

$$\underset{Z}{\text{minimize}} \ \frac{1}{2}\|\Omega \circ (X - Z)\|_F^2 + \lambda \|Z\|_*$$

softImpute

0. Initialize Z in some way (eg 0s or warm start).

Repeat till convergence.

1. Compute $Z^{\text{new}} \leftarrow \mathbf{S}_\lambda \left(\Omega \circ X + (1 - \Omega) \circ Z^{\text{old}}\right)$.

This is an MM algorithm for solving the nuclear-norm regularized problem
Can solve this for a decreasing sequence of λs
$\lambda_1 > \lambda_2 > \cdots \lambda_j > \lambda_{j+1} > \cdots$ using warm starts.

softImpute : Computational Bottleneck

At kth iteration of step 1. need to solve the soft SVD problem

$$Z_{k+1} = \arg\min_{Z} \frac{1}{2}\|\Omega \circ X + (1-\Omega) \circ Z_k - Z\|_F^2 + \lambda\|Z\|_*$$

Soft SVD requires (low-rank) SVD of *completed* matrix after k iterations:

$$\widehat{X}_k = \Omega \circ X + (1-\Omega) \circ Z_k$$

For large applications (Netflix), this matrix is huge!
Trick:

$$\underset{}{\Omega \circ X + (1-\Omega) \circ Z_k} \quad = \quad \underset{\textit{Sparse}}{\Omega \circ (X - Z_k)} \quad + \quad \underset{\textit{Low Rank}}{Z_k}$$

Computational tricks in softImpute

- Anticipate rank of $\widehat{Z}_{\lambda_{j+1}}$ based on rank of $\widehat{Z}_{\lambda_j}$, erring on generous side.
- Compute low-rank SVD of $\widehat{X}_k$ using orthogonal QR iterations with Reitz acceleration (Stewart, 1969, H, Mazumder, Lee and Zadeh 2015 [JMLR]).
- Iterations require left and right multiplications $U^\top \widehat{X}_k$ and $\widehat{X}_k V$. Ideal for *Sparse + Low-Rank* structure.
- Warm starts: $\mathbf{S}_\lambda(\widehat{X}_k)$ provides excellent warm starts (U and V) for $\mathbf{S}_\lambda(\widehat{X}_{k+1})$. Likewise $\widehat{Z}_{\lambda_j}$ for $\widehat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m+n)\cdot r + |\Omega|]$.

Computational tricks in softImpute

- Anticipate rank of $\widehat{Z}_{\lambda_{j+1}}$ based on rank of $\widehat{Z}_{\lambda_j}$, erring on generous side.
- Compute low-rank SVD of $\widehat{X}_k$ using orthogonal QR iterations with Reitz acceleration (Stewart, 1969, H, Mazumder, Lee and Zadeh 2015 [JMLR]).
- Iterations require left and right multiplications $U^\top \widehat{X}_k$ and $\widehat{X}_k V$. Ideal for *Sparse + Low-Rank* structure.
- Warm starts: $\mathbf{S}_\lambda(\widehat{X}_k)$ provides excellent warm starts (U and V) for $\mathbf{S}_\lambda(\widehat{X}_{k+1})$. Likewise $\widehat{Z}_{\lambda_j}$ for $\widehat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m+n)\cdot r + |\Omega|]$.

Computational tricks in softImpute

- Anticipate rank of $\widehat{Z}_{\lambda_{j+1}}$ based on rank of $\widehat{Z}_{\lambda_j}$, erring on generous side.
- Compute low-rank SVD of $\widehat{X}_k$ using orthogonal QR iterations with Reitz acceleration (Stewart, 1969, H, Mazumder, Lee and Zadeh 2015 [JMLR]).
- Iterations require left and right multiplications $U^\top \widehat{X}_k$ and $\widehat{X}_k V$. Ideal for *Sparse + Low-Rank* structure.
- Warm starts: $\mathbf{S}_\lambda(\widehat{X}_k)$ provides excellent warm starts (U and V) for $\mathbf{S}_\lambda(\widehat{X}_{k+1})$. Likewise $\widehat{Z}_{\lambda_j}$ for $\widehat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m+n)\cdot r + |\Omega|]$.

Computational tricks in softImpute

- Anticipate rank of $\widehat{Z}_{\lambda_{j+1}}$ based on rank of $\widehat{Z}_{\lambda_j}$, erring on generous side.
- Compute low-rank SVD of $\widehat{X}_k$ using orthogonal QR iterations with Reitz acceleration (Stewart, 1969, H, Mazumder, Lee and Zadeh 2015 [JMLR]).
- Iterations require left and right multiplications $U^\top \widehat{X}_k$ and $\widehat{X}_k V$. Ideal for *Sparse + Low-Rank* structure.
- Warm starts: $\mathbf{S}_\lambda(\widehat{X}_k)$ provides excellent warm starts (U and V) for $\mathbf{S}_\lambda(\widehat{X}_{k+1})$. Likewise $\widehat{Z}_{\lambda_j}$ for $\widehat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m+n)\cdot r + |\Omega|]$.

Computational tricks in softImpute

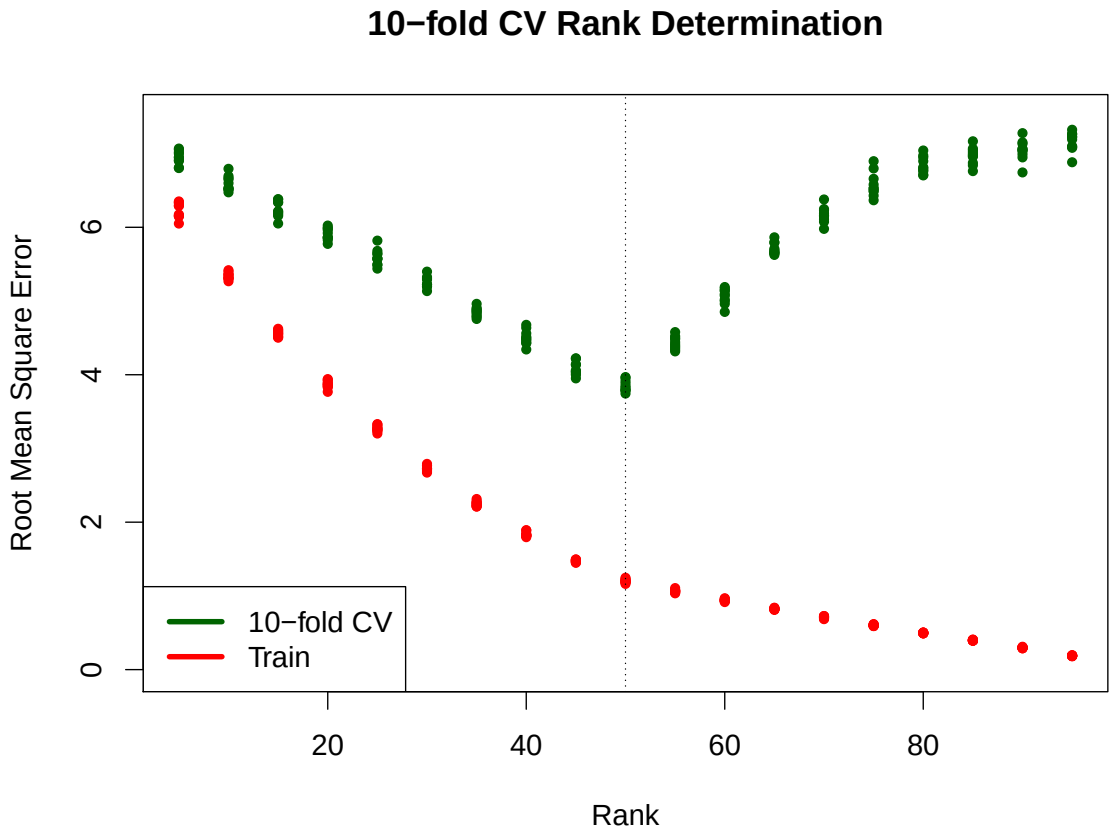
- Anticipate rank of $\widehat{Z}_{\lambda_{j+1}}$ based on rank of $\widehat{Z}_{\lambda_j}$, erring on generous side.
- Compute low-rank SVD of $\widehat{X}_k$ using orthogonal QR iterations with Reitz acceleration (Stewart, 1969, H, Mazumder, Lee and Zadeh 2015 [JMLR]).
- Iterations require left and right multiplications $U^\top \widehat{X}_k$ and $\widehat{X}_k V$. Ideal for *Sparse + Low-Rank* structure.
- Warm starts: $\mathbf{S}_\lambda(\widehat{X}_k)$ provides excellent warm starts (U and V) for $\mathbf{S}_\lambda(\widehat{X}_{k+1})$. Likewise $\widehat{Z}_{\lambda_j}$ for $\widehat{Z}_{\lambda_{j+1}}$.
- Total cost per iteration $O[(m+n)\cdot r + |\Omega|]$.

softImpute on Netflix problem

rank	time (hrs)	RMSE	% Improvement
42	1.36	0.9622	-1.1
66	2.21	0.9572	-0.6
81	2.83	0.9543	-0.3
Cinematch		0.9514	0
95	3.27	0.9497	1.8
120	4.40	0.9213	3.2
.	.	.	
.	.	.	
Winning Goal		0.8563	10

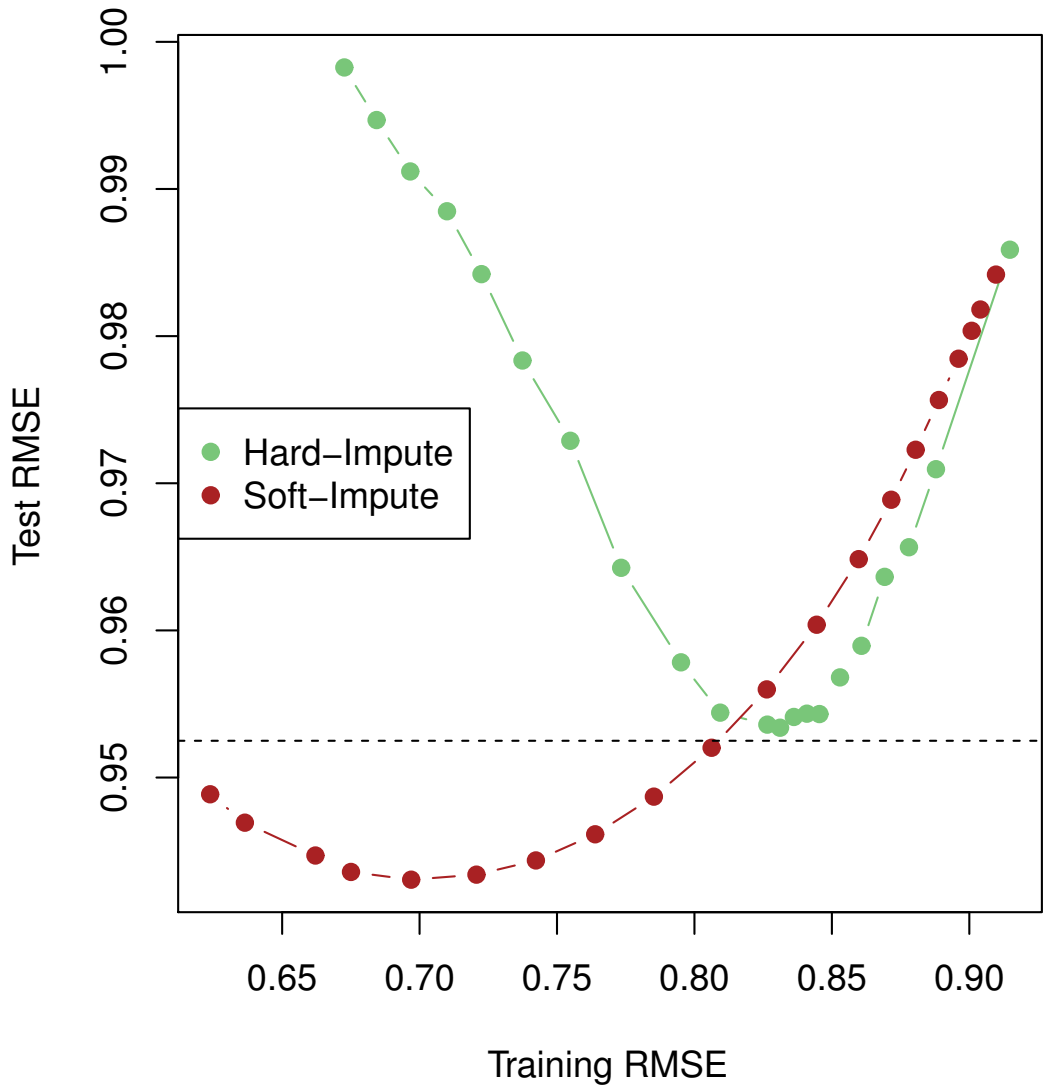
state-of-the-art convex solvers do not scale to this size

Example: choosing a good rank for SVD

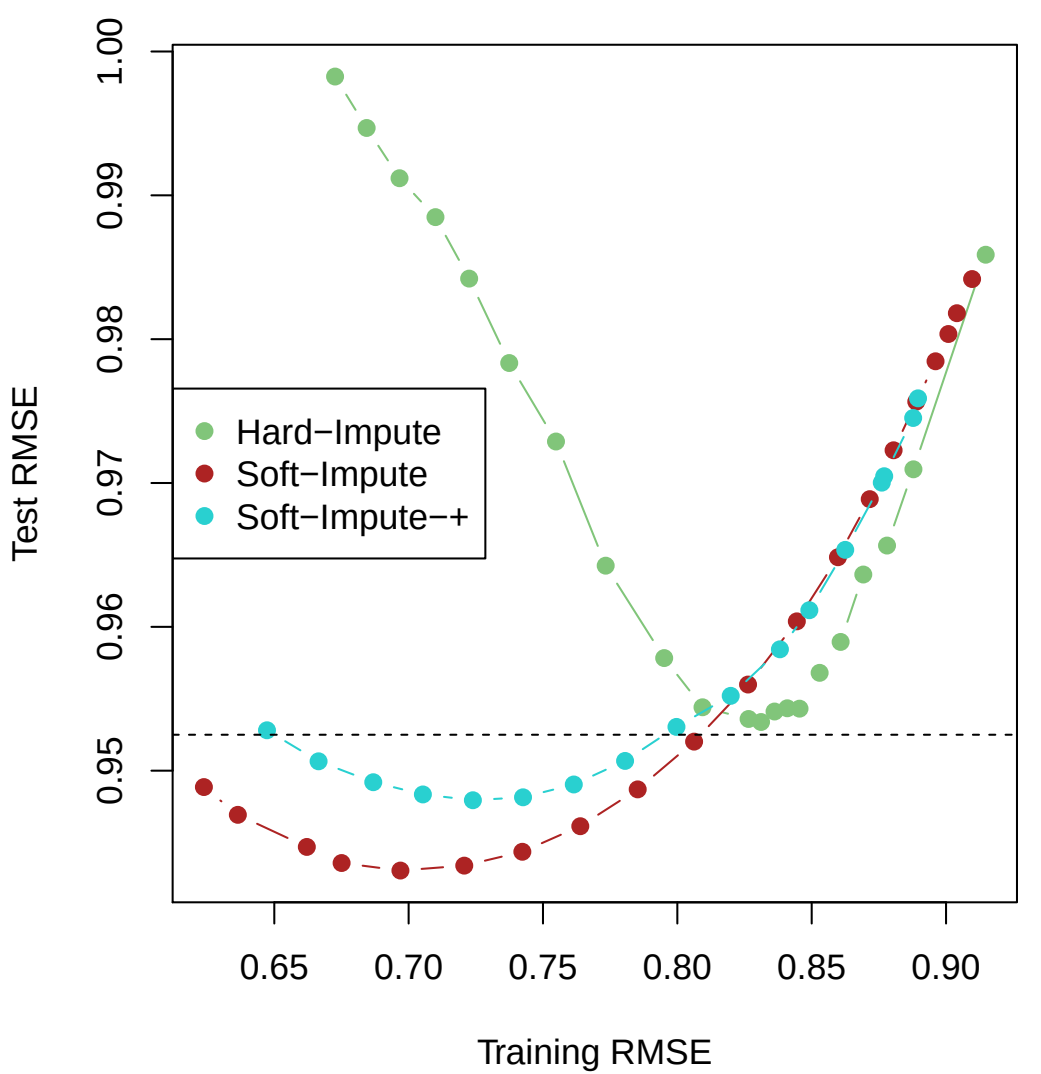


Truth is 200×100 rank-50 matrix plus noise (SNR 3).
Randomly omit 10% of entries, and then predict using solutions from softImpute or hardImpute.

softImpute beats hardImpute on Netflix



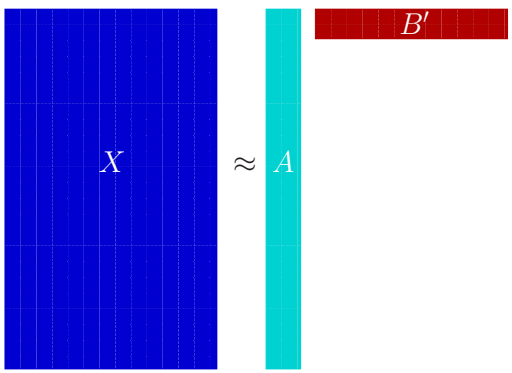
softImpute beats debiased softImpute on Netflix



Alternating Least Squares

Consider rank-r approximation $Z = A_{m\times r}B_{n\times r}^\top$, and solve

$$\underset{A,B}{\text{minimize}} \ \|\Omega \circ (X - AB^\top)\|_F^2 + \lambda(\|A\|_F^2 + \|B\|_F^2)$$



- Regularized SVD (Srebro et al 2003, Simon Funk)
- Not convex, but *bi-convex*: alternating ridge regression

Lemma (Srebro et al 2005, Mazumder et al 2010)
For any matrix W, the following holds:

$$\|W\|_* = \min_{A,B:\ W=AB^\top} \frac{1}{2}\left(\|A\|_F^2 + \|B\|_F^2\right).$$

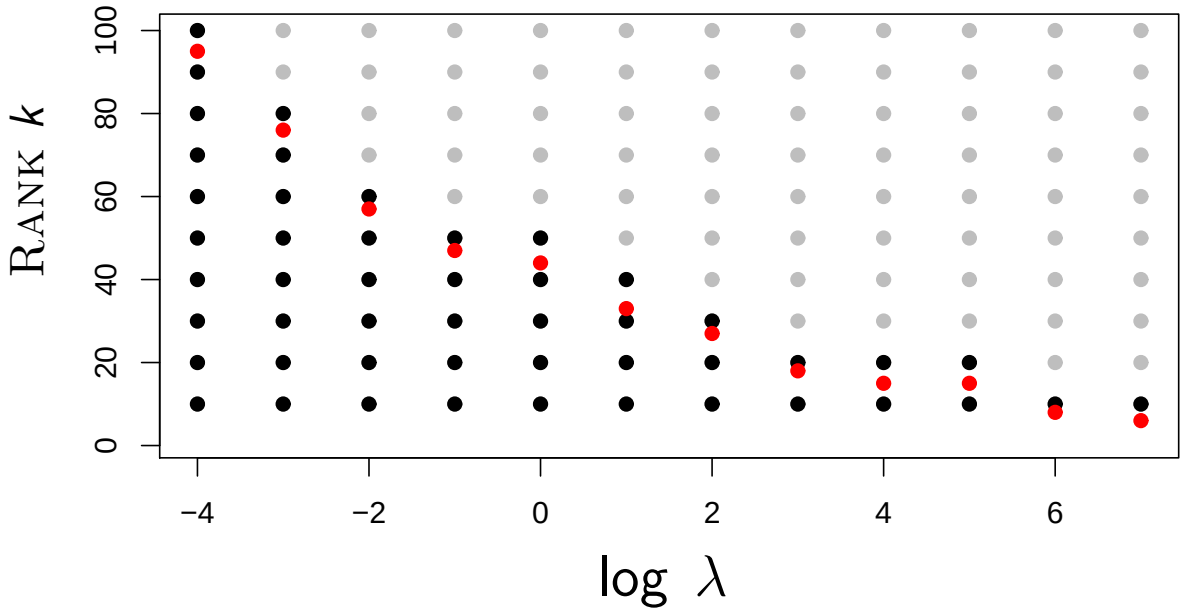
If $\text{rank}(W) = k \le \min\{m, n\}$, then the minimum above is attained at a factor decomposition $W = A_{m\times k}B_{n\times k}^\top$.

Connections between ALS and softImpute

$$\text{ALS}: \quad \underset{A_{n\times r}, B_{m\times r}}{\text{minimize}} \ \frac{1}{2}\|\Omega \circ (X - AB^\top)\|_F^2 + \frac{\lambda}{2}(\|A\|_F^2 + \|B\|_F^2)$$

$$\text{softImpute}: \quad \underset{Z}{\text{minimize}} \ \frac{1}{2}\|\Omega \circ (X - Z)\|_F^2 + \lambda\|Z\|_*$$

- Solution-space of ALS contains solutions of softImpute.
- For *large* rank r: ALS $\equiv$ softImpute.



Synthesis and New Approach

- ALS is slower than SOFTIMPUTE — factor of 10.
- ALS requires guesswork for rank, and does not return a definitive low-rank solution.
- SOFTIMPUTE requires a low-rank SVD at each iteration. Typically iterative QR methods are used, exploiting problem structure and warms starts.

Idea: combine softImpute and ALS

- Leads to algorithm more efficient than SOFTIMPUTE
- Scales naturally to larger problems using parallel/multicore programming
- Suggests efficient algorithm for low-rank SVD for complete matrices

Nuclear-norm and ALS results

Consider fully observed $X_{n\times m}$.

$$\text{Nuclear}: \quad \underset{\text{rank}(Z)\le r}{\text{minimize}} \ \frac{1}{2}\|X-Z\|_F^2+\lambda\|Z\|_*$$

$$\text{ALS}: \quad \underset{A_{n\times r},B_{m\times r}}{\text{minimize}} \ \frac{1}{2}\|X-AB^\top\|_F^2+\frac{\lambda}{2}(\|A\|_F^2+\|B\|_F^2)$$

The solution to Nuclear is

$$Z=U_rD_*V_r^\top,$$

where U_r and V_r are first r left and right singular vectors of X, and

$$D_*=\text{diag}[(\sigma_1-\lambda)_+,\ldots,(\sigma_r-\lambda)_+]$$

A solution to ALS is

$$A=U_rD_*^{\frac{1}{2}} \text{ and } B=V_rD_*^{\frac{1}{2}}$$

Consequences of new nuclear-norm / ALS connections

For SVD of fully observed matrix:

- Can solve reduced-rank SVD by alternating ridge regressions.
- At each iteration, re-orthogonalization as in usual QR iterations (for reduced-rank SVD) means ridge regression is a simple matrix multiply, followed by column scaling.
- Ridging speeds up convergence, and focuses accuracy on leading dimensions.
- Solution delivers a reduced-rank SVD.

For matrix completion:

- Combine SVD calculation and imputation in softImpute.
- Leads to a faster algorithm that can be distributed to multiple cores for storage and computation efficiency.

Consequences of new nuclear-norm / ALS connections

For SVD of fully observed matrix:

- Can solve reduced-rank SVD by alternating ridge regressions.
- At each iteration, re-orthogonalization as in usual QR iterations (for reduced-rank SVD) means ridge regression is a simple matrix multiply, followed by column scaling.
- Ridging speeds up convergence, and focuses accuracy on leading dimensions.
- Solution delivers a reduced-rank SVD.

For matrix completion:

- Combine SVD calculation and imputation in softImpute.
- Leads to a faster algorithm that can be distributed to multiple cores for storage and computation efficiency.

softImpute//

Back to matrix imputation.

1. Initialize $U_{n\times r}$, $V_{m\times r}$ orthogonal, $D_{r\times r} > 0$ diagonal, and $A = UD$, $B = VD$.
2. Given U and D and hence $A = UD$, update B:

   2.a Compute current imputation:

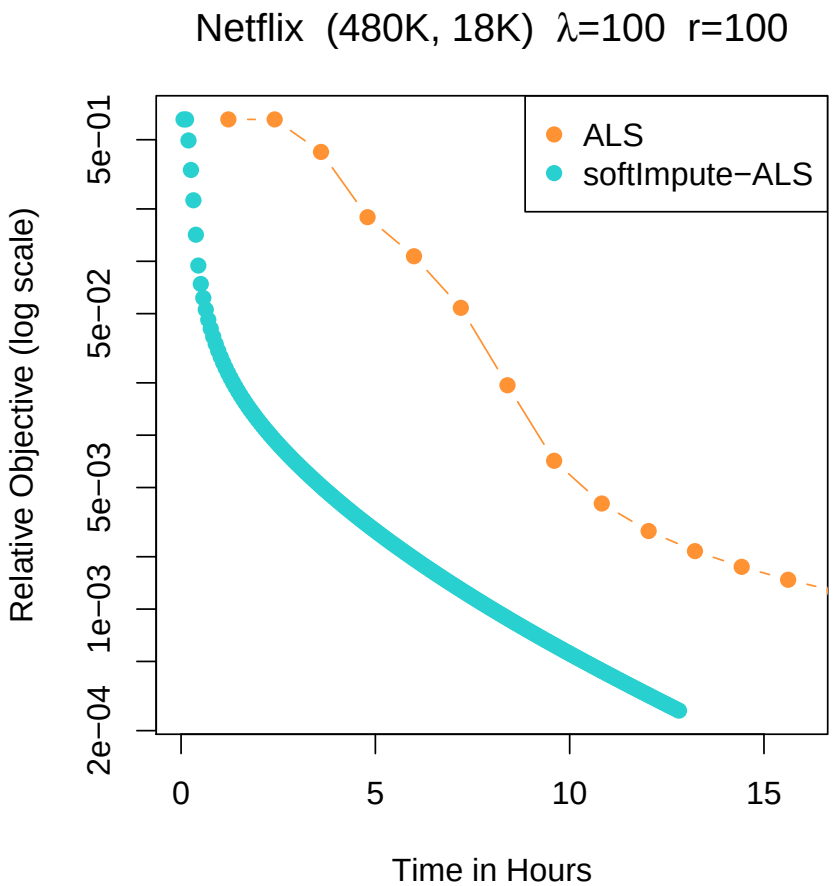
$$\begin{aligned} X^* &= \Omega \circ X + (1-\Omega) \circ (AB^\top) \\ &= \Omega \circ (X - AB^\top) + UD^2V^\top \end{aligned}$$

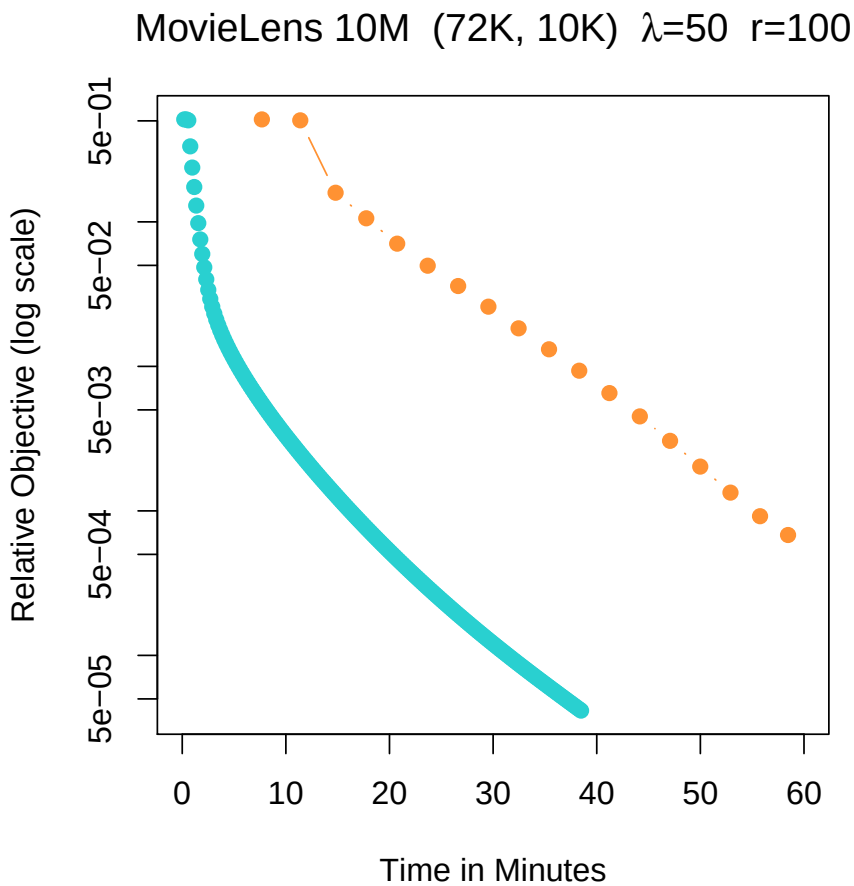
   2.b Ridge regression of X^* on A:

$$\begin{aligned} B^\top &\leftarrow (D^2 + \lambda I)^{-1} DU^\top X^* \\ &= D_1 U^\top \left[\Omega \circ (X - AB^\top)\right] + D_2 V^\top \end{aligned}$$

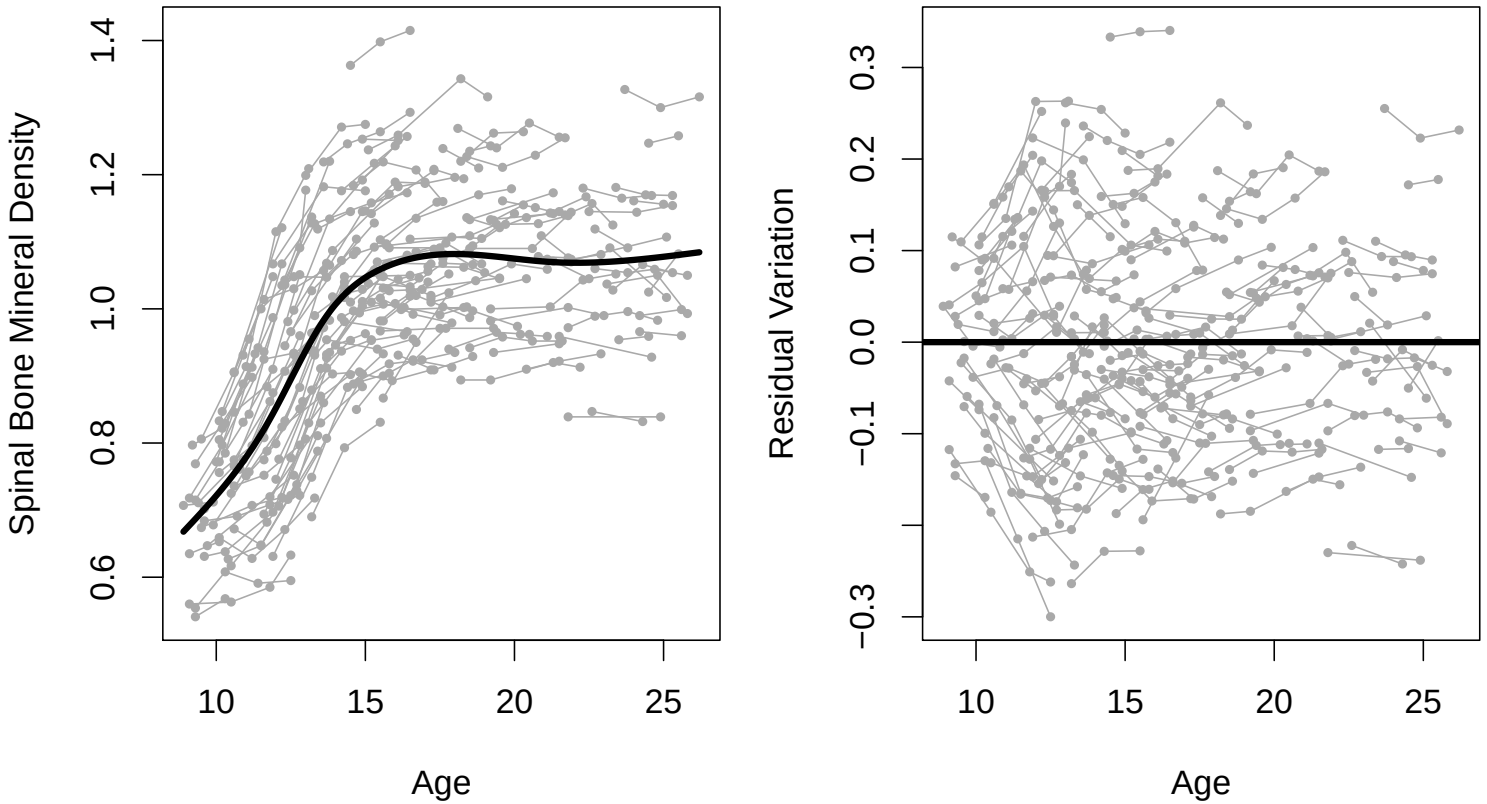
   2.c Reorthogonalize and update V, D and U via SVD of BD.
3. Given V and D and $B = VD$, update A in similar fashion.
4. At convergence, U and V provides SVD of X^*, and hence $\mathbf{S}_\lambda(X^*)$, which *cleans up* the rank of the solution.

Timing Comparisons





Functional Principal Components

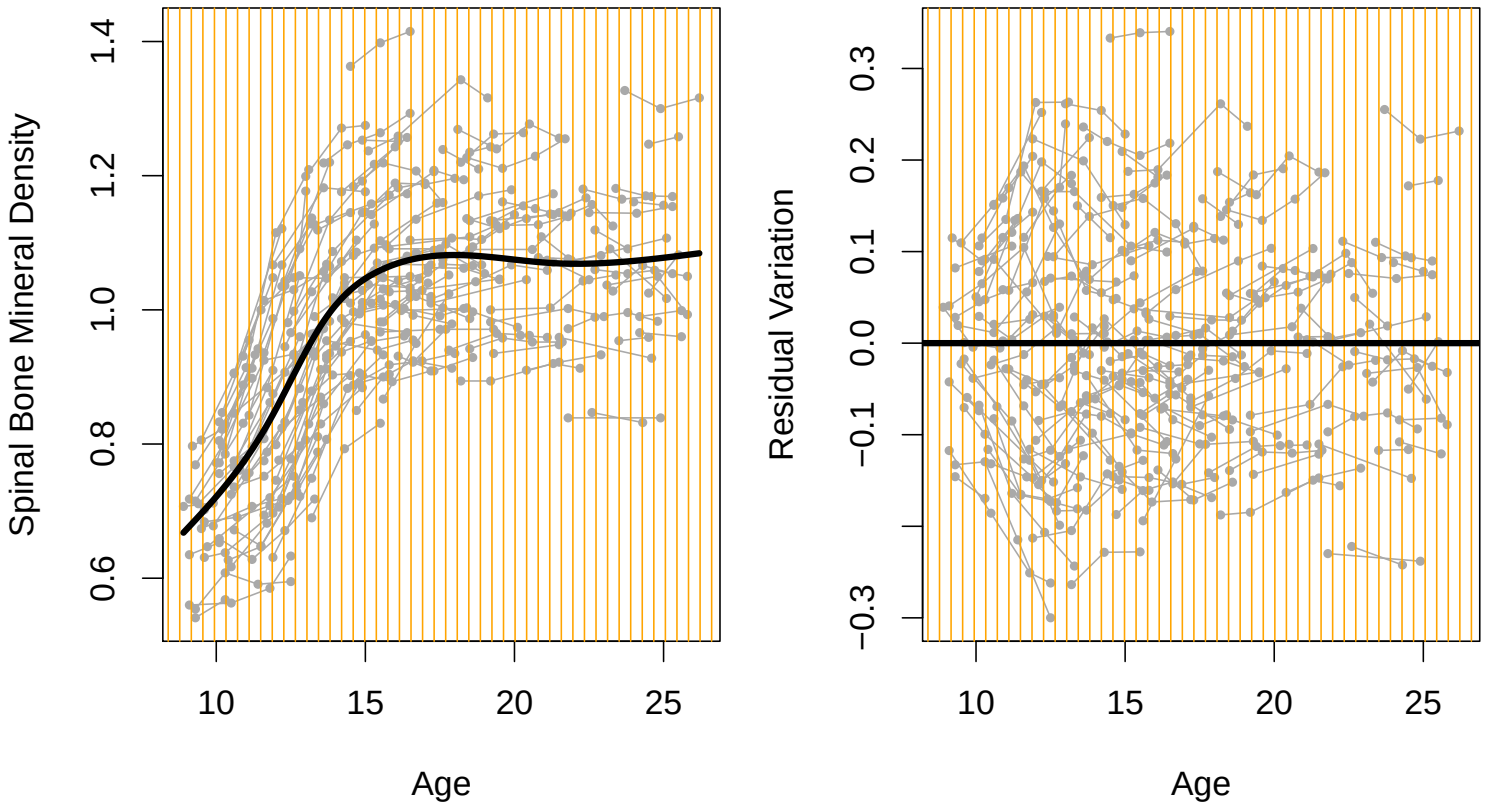


Idea:

- Approximate data to a fine age grid. Results in a matrix with each row a subject, and many missing entries.
- Modify softImpute so rows are smooth in age.

Similar approach in Fithian and Mazumder (SS 2018, "Side Information" paper)

Functional Principal Components

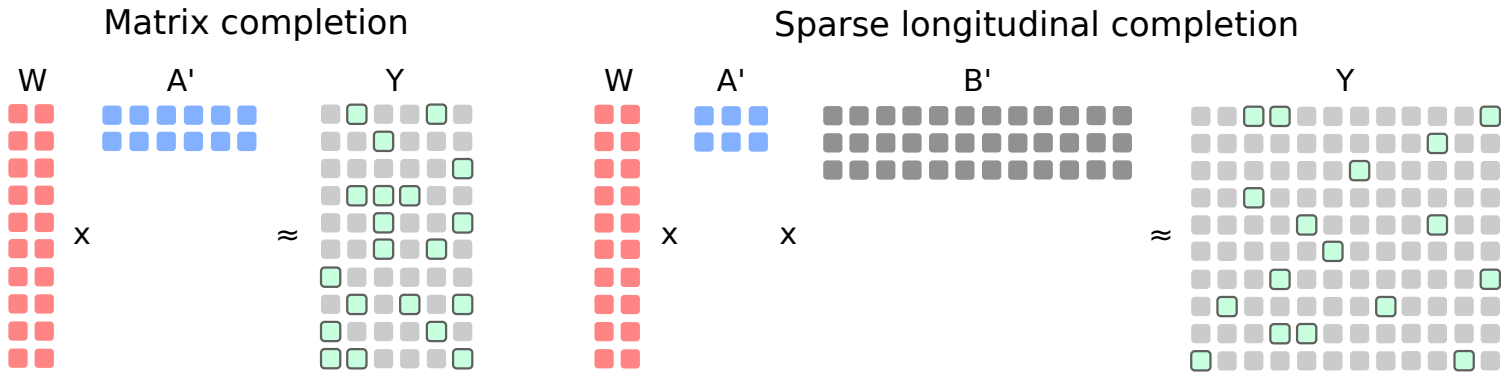


Idea:

- Approximate data to a fine age grid. Results in a matrix with each row a subject, and many missing entries.
- Modify softImpute so rows are smooth in age.

Similar approach in Fithian and Mazumder (SS 2018, "Side Information" paper)

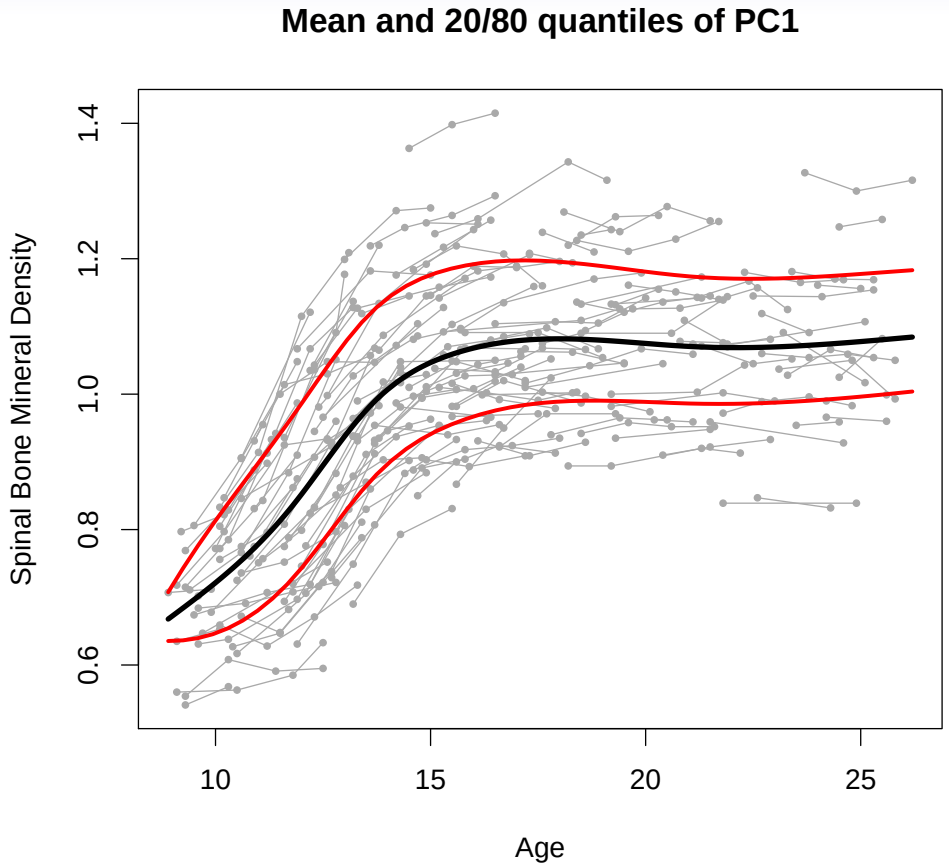
Functional softImpute



$$\underset{Z}{\text{minimize}} \quad \frac{1}{2}\|\Omega \circ (X - ZB^\top)\|_F^2 + \lambda\|Z\|_*$$

where low-rank solution $Z = WA^\top$, and B is orthonormal basis of smooth (spline) functions in the index (age).

1. $\widehat{X} \leftarrow \Omega \circ X + (1 - \Omega) \circ \ (Z^{\text{old}} B^\top)$
2. $Z^{\text{new}} \leftarrow \mathbf{S}_\lambda(\widehat{X}B)$



Package fcomplete by Lukasz Kidzinski fits functional PCA and regression models to data of this sort.
Also implements related EM algorithms for fPCA by James, H, and Sugar (Biometrika, 2000)



Application: modeling species distributions

We have species presence/absence or counts y_{ij} for $j = 1, \ldots, m$ species at $i = 1, \ldots, n$ locations.

Generalized Linear Latent Variable Model (GLLVM):

$$\begin{aligned} y_{ij}|\mu_{ij} &\sim \mathcal{F}(\mu_{ij}, \phi_j) \quad \text{Poisson, binomial,...} \\ g(\mu_{ij}) &= \beta_{0j} + x_i^\top \beta_j + \lambda_j^\top u_i \\ u_i &\sim N(0, I) \end{aligned}$$

Application: modeling species distributions

We have species presence/absence or counts y_{ij} for $j = 1, \ldots, m$ species at $i = 1, \ldots, n$ locations.
Generalized Linear Latent Variable Model (GLLVM):

$$\begin{aligned} y_{ij}|\mu_{ij} &\sim \mathcal{F}(\mu_{ij}, \phi_j) \quad \text{Poisson, binomial,...} \\ g(\mu_{ij}) &= \beta_{0j} + x_i^\top \beta_j + \lambda_j^\top u_i \\ u_i &\sim N(0, I) \end{aligned}$$

Current likelihood methods are slow, and do not scale well.
With Lukasz Kidzinski, Francis Hui and David Warton, we adapt ideas in SOFIMPUTE to fit models at scale (48K locations, 5K species) using penalized quasi likelihood.

Application: modeling species distributions

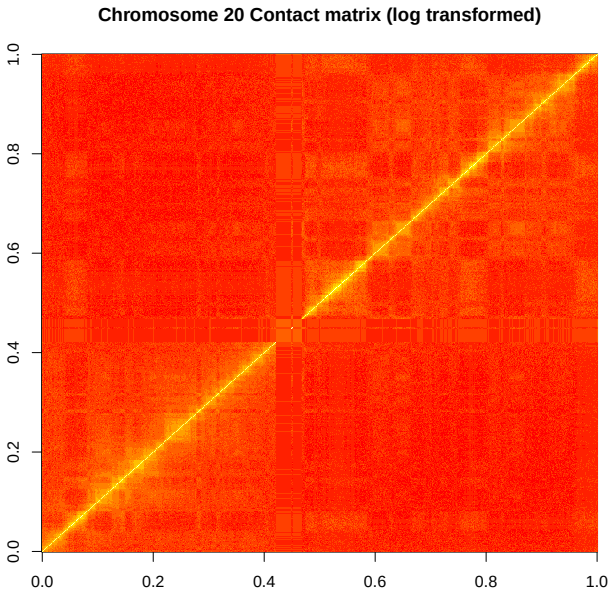
We have species presence/absence or counts y_{ij} for $j = 1, \ldots, m$ species at $i = 1, \ldots, n$ locations.
Generalized Linear Latent Variable Model (GLLVM):

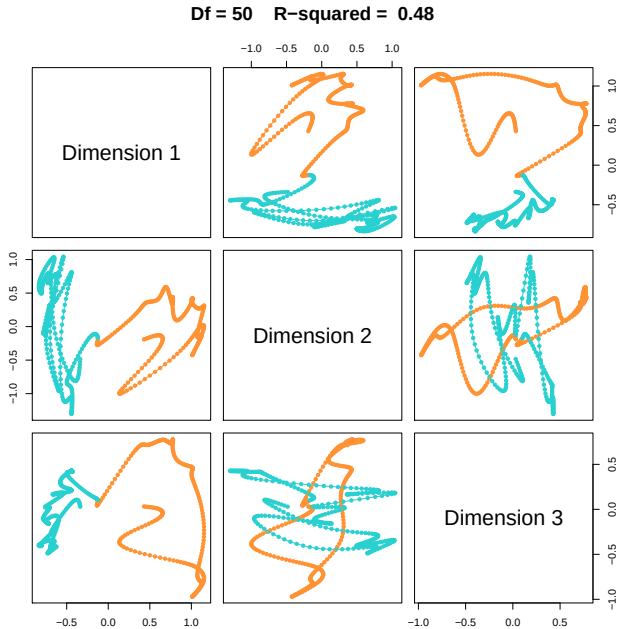
$$\begin{aligned} y_{ij}|\mu_{ij} &\sim \mathcal{F}(\mu_{ij}, \phi_j) \quad \text{Poisson, binomial,...} \\ g(\mu_{ij}) &= \beta_{0j} + x_i^\top \beta_j + \lambda_j^\top u_i \\ u_i &\sim N(0, I) \end{aligned}$$

Current likelihood methods are slow, and do not scale well.
With Lukasz Kidzinski, Francis Hui and David Warton, we adapt ideas in SOFIMPUTE to fit models at scale (48K locations, 5K species) using penalized quasi likelihood.

Related work:
Robin, Josse, Moulines and Sardy (ArXiv 2017)
Fithian and Mazumder (*SS* 2018)
2019 Ph.D thesis of Genevieve Robin (adv. J. Josse & E Moulines)
Lin and Breslow (Jasa 1996)

Application: chromatin folding

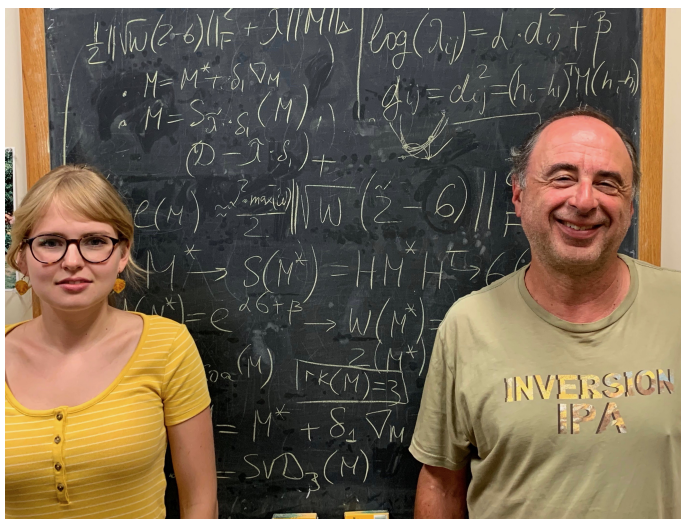




Use approach based on principal curves in context of matrix Poisson model

Weighted version of softImpute plays a key role

with Elena Tuzhilina and Mark Segal



Software Implementations

- `softImpute` package in R. Can deal with large sparse complete matrices, or large matrices with many missing entries (ie Netflix or bigger). Includes row and column centering and scaling options.
- Spark cluster-programming. Uses distributed computing and chunking. Can deal with very large problems (e.g. $10^7 \times 10^7$, 139 secs per iteration). See `http://git.io/sparkfastals` with documentation in Scala.
- `fcomplete` package for fitting sparse functional PCA and regression models. `github: kidzik/fcomplete`

Software Implementations

- `softImpute` package in R. Can deal with large sparse complete matrices, or large matrices with many missing entries (ie Netflix or bigger). Includes row and column centering and scaling options.
- Spark cluster-programming. Uses distributed computing and chunking. Can deal with very large problems (e.g. $10^7 \times 10^7$, 139 secs per iteration). See `http://git.io/sparkfastals` with documentation in Scala.
- `fcomplete` package for fitting sparse functional PCA and regression models. `github: kidzik/fcomplete`

Thank you for attending!