Multivariate Statistical Methods for Big Data Analysis and Process Improvement

Instructor: Dr. Brandon Corbett

Lecture 10 for ChE 765 | Sep 767, McMaster University

Agenda:

Variable transformations:

1. Multi-block methods
2. Other data pre-processing

Multiblock methods

The main concept

Divide your variables into blocks to get

- better model interpretation
- easier monitoring and improved fault detection

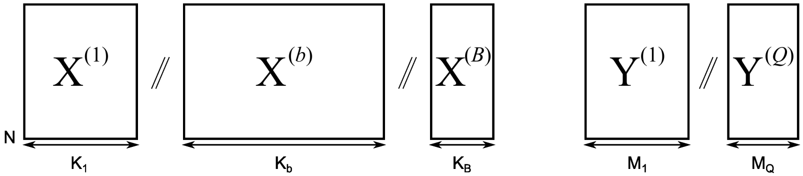
Why do this?: we'd like to understand the relationships between several groups of possibly related datasets

Sometimes called "data fusion".

References

- Original concept: Wold *et al.*, 1987 conference paper
- Improved fault detection: MacGregor *et al.*, AIChE Journal
- Equivalence of MBPCA and PBPLS to PCA and PLS (very important paper): Westerhuis, Kourti and MacGregor
- Process monitoring example with MB methods: Qin *et al.*
- Good overview of all multiblock methods: Smilde, Westerhuis and de Jong, 2003

Notation



- Multiple **X** and **Y** blocks are available
- There is only one consistent dimension: N = observations
- We will only consider the case of one **Y** block ($M_1 = M$)
  - **Y** will contain the usual quality variables
- We can have in $\mathbf{X}^{(b)}$ for example:
  - raw material properties (e.g. one block per material)
  - NIR or UV-VIS spectra from each observation
  - Unfolded batch data
  - Measurements from each unit operation

Key point: you can have duplicated variables between blocks

Terminology and concepts

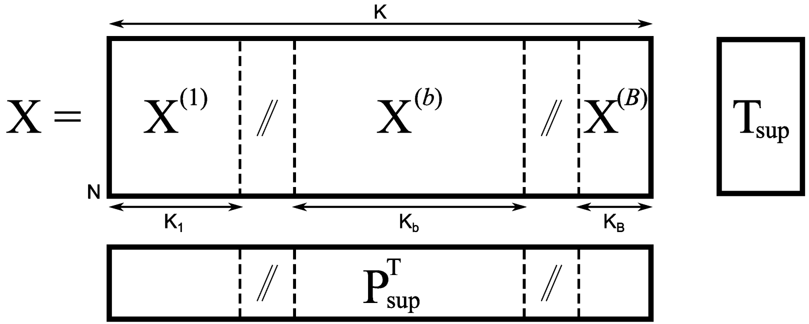
- Only have **X** blocks: multiblock PCA
- Add one or more **Y** blocks: then it becomes multiblock PLS

- Each block has: scores, loadings, SPE, T^2, weights, VIP, R^2

- We also have a “super-level” or “super-model” that summarizes the blocks

SUM-PCA approach

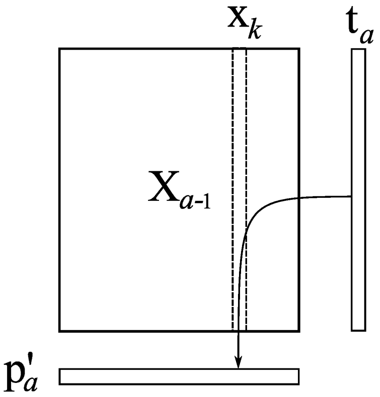
Crude approach: push all blocks together and build PCA model.

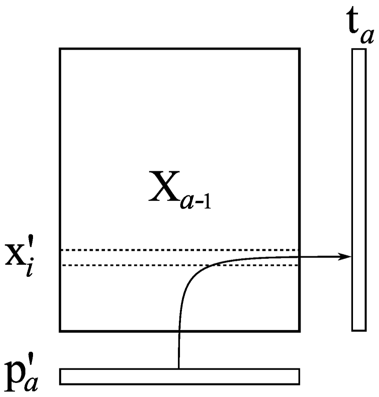


- Investigate loadings, R^2, *etc* separately for each block
- Block loadings, $P^{(b)}$, will not be orthogonal
- The super scores (the usual PCA scores) simply explain variation for entire **X**
- No guarantee that each block will contribute to superscores

NIPALS review

Before we proceed, let's recap the NIPALS algorithm for PCA





Loadings are regression coefficients (slopes) when regressing columns in $\mathbf{X}$ onto $\mathbf{t}_a$

Scores are regression coefficients (slopes) when regressing rows in $\mathbf{X}$ onto $\mathbf{p}_a$

Consensus PCA (CPCA)

Left blank for you to draw the diagram

Consensus PCA steps

1. Let $t_a^{(s)}$ be any column from any block
2. Regress column from $\mathbf{X}_a^{(b)}$ onto $\mathbf{t}_a^{(s)}$ to obtain block loadings

$$\mathbf{p}_a^{(b)} = \mathbf{X}^{(b)T}\mathbf{t}_a^{(s)} / \mathbf{t}_a^{(s)T}\mathbf{t}_a^{(s)}$$

3. Normalize $\mathbf{p}_a^{(b)}$ to unit length
4. Calculate block's score: $\mathbf{t}_a^{(b)} = \mathbf{X}^{(b)}\mathbf{p}_a^{(b)} \cdot \frac{1}{\sqrt{K_b}}$
   - weight $\sqrt{K_b}$ prevents blocks with many terms (variables) in the above linear combination from creating large score values, $\mathbf{t}_a^{(b)}$
5. Assemble block scores: $\mathbf{T}_a^{[s]} = \left[\mathbf{t}_a^{(1)} \ \ldots \ \mathbf{t}_a^{(b)} \ \ldots \ \mathbf{t}_a^{(B)}\right]$

Consensus PCA steps

6. Regress columns in $\mathbf{T}_a^{[s]}$ onto the superscore, $\mathbf{t}_a^{(s)}$ to calculate the super-level's loading:

$$\begin{array}{rcl} \mathbf{p}_a^{[s]} & = & \mathbf{T}_a^{[s]\,T}\mathbf{t}_a^{(s)} / \left(\mathbf{t}_a^{(s)\,T}\mathbf{t}_a^{(s)}\right) \\ & & (B \times N)(N \times 1) \end{array}$$

7. Normalize $\mathbf{p}_a^{[s]}$ to unit length
8. Regress rows in $\mathbf{T}_a^{[s]}$ onto $\mathbf{p}_a^{[s]}$ to get the super-scores $\mathbf{t}_a^{(s)}$:

$$\begin{array}{rcl} \mathbf{t}_a^{(s)} & = & \mathbf{T}_a^{[s]}\mathbf{p}_a^{[s]} / \left(\mathbf{p}_a^{[s]\,T}\mathbf{p}_a^{[s]}\right) \\ & & (N \times B)(B \times 1) \end{array}$$

denominator is usually $= 1.0$

9. Not converged? return back to step 2.
10. Converged? deflate each block *with the superscore*

$$\mathbf{X}_a^{(b)} = \mathbf{X}_a^{(b)} - \mathbf{t}_a^{(s)}\mathbf{p}_a^{(b)\,T}$$

Consensus PCA (CPCA)

- $\mathbf{t}_a^{(s)}\mathbf{p}_a^{(b)}$ = block prediction from the *superscore*, $\mathbf{t}_a^{(s)}$, not the block's score
- $\mathbf{t}_a^{(s)}$ was calculated from the assembled scores, $\mathbf{T}_a^{[s]}$
- $\mathbf{t}_a^{(s)}$ is just a weighted sum of these block scores (step 8): called the *consensus score*
- **Each entry in the superloading shows how much of block *b* is used in the consensus score**
- If a block behaves differently from the others, then its entry in $\mathbf{p}_a^{(s)}$ will be small
- Deflation by $\mathbf{t}_a^{(s)}$ removes the superscore information, not the block-score information.
  - We get non-orthogonal block scores, but orthogonal superscores

Computational simplification

Westerhuis, Kourti and MacGregor (1998) showed we don't need to calculate CPCA as just described.

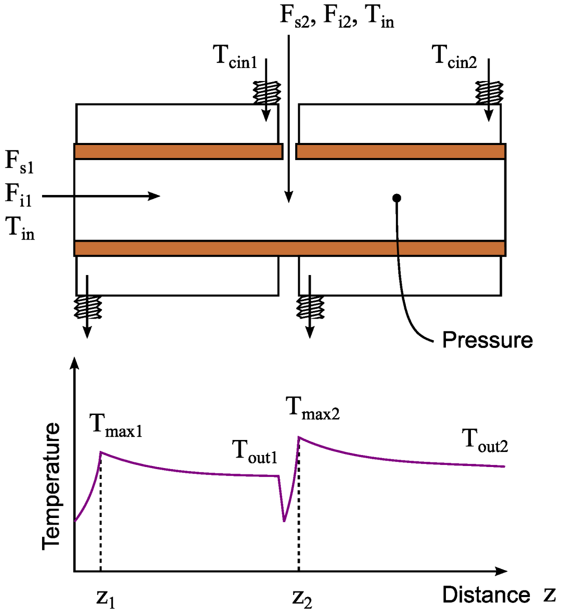
Much easier approach:

- Preprocess the data from each block as normal
- Post divide each block by $\sqrt{K_b}$ and assemble:

$$\mathbf{X} = \left[\frac{\mathbf{X}^{(1)}}{\sqrt{K_1}},\ \frac{\mathbf{X}^{(2)}}{\sqrt{K_2}},\ \ldots,\ \frac{\mathbf{X}^{(B)}}{\sqrt{K_B}} \right]$$

  - Same idea as block-scaling (covered earlier in the course)
- Calculate PCA in the usual way on $\mathbf{X}$ to obtain:
  - scores will be identical to CPCA super scores, $\mathbf{t}_1^{(s)}, \mathbf{t}_2^{(s)}, \ldots, \mathbf{t}_A^{(s)}$
  - then follow steps 2, 3, 4, 5 and 6 from above
  - results will be identical to the full approach

In-class example



In-class example

Load the LDPE data set and create a 2-block PCA model:

1. "Zone 1" block
   - Inlet temperature
   - Pressure
   - All other variables ending in "1"
2. 'Zone 2" block
   - Inlet temperature
   - Pressure
   - All other variables ending in "2"

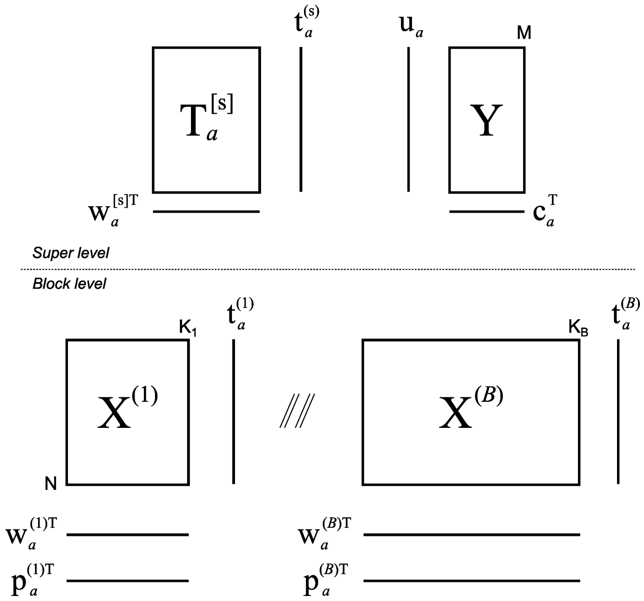
Build a multiblock CPCA model, using cross-validation to determine *A*:

- examine scores for each block, and the superblock
- examine the loadings bi-plot for each block
- example the SPE time-series for each block and the superblock

What can go into each block?

- Raw material properties
  - have one block per raw material
- NIR or UV-VIS spectra ($K^{(b)}$ will be large)
- Unfolded batch trajectories
- Features extracted from batch trajectories
- Data from sequential operations
  - data from each step/operation/phase in its own block
  - could be hard to ensure consistency from row to row
- Large PCA and PLS models. E.g. petroleum refinery
  - distillation column's data
  - fractionator's data
  - FCCU data
- Judges: one block per judge
  - each judge block contains the same columns (attributes)
- Lagged variables
  - e.g. a variable and all its lag per block
  - or, all variables at a particular lag in each block

Multiblock PLS (MBPLS) concept



MBPLS concept

We won't go through the detailed arrow pushing diagrams:

1. Start with an initial guess for $\mathbf{u}_a$
2. Perform a CPCA cycle through all the $\mathbf{X}^{(b)}$ blocks and this $\mathbf{u}_a$
3. Assemble each block's scores, $\mathbf{t}_a^{(b)}$, into $\mathbf{T}_a^{[s]} = \left[\mathbf{t}_a^{(1)} \ \ldots \ \mathbf{t}_a^{(B)}\right]$
4. Do a single NIPALS cycle for PLS between $\mathbf{T}_a^{[s]}$ and $\mathbf{Y}$ for
   - super scores, $\mathbf{t}_a^{(s)}$
   - super weights, $\mathbf{w}_a^{[s]}$, a $B \times 1$ vector
   - **Y**-space loadings: $\mathbf{c}_a$, a $M \times 1$ vector
   - **Y**-space scores: $\mathbf{u}_a$
5. Repeat from step 2 until convergence for the a^{th} component
6. Then deflate ... (next slide)

Once all components calculated, predict $\hat{\mathbf{Y}} = \mathbf{t}_1^{(s)}\mathbf{c}_1 + \ldots + \mathbf{t}_A^{(s)}\mathbf{c}_A$

MBPLS deflation

There are 2 choices to deflate each block:

1. using the block's own score and loading

$$\mathbf{X}^{(b)} = \mathbf{X}^{(b)} - \mathbf{t}_a^{(b)}\mathbf{p}_a^{(b)T}$$

   - induces orthogonal scores and loadings at the block level
   - super scores, $\mathbf{t}_a^{(s)}$, will not be orthogonal

2. using the super score and the block's loading

$$\mathbf{X}^{(b)} = \mathbf{X}^{(b)} - \mathbf{t}_a^{(s)}\mathbf{p}_a^{(b)T}$$

   - block level scores and loadings not orthogonal
   - super scores are orthogonal

Using the MBPLS model in the future

1. Center and scale new data, $\mathbf{x}_{\text{new}}^{(b)}$, according to each block's preprocessing
2. Calculate block score $= t_{a,\text{new}}^{(b)} = \mathbf{x}_{\text{new}}^{(b)T}\mathbf{w}_a^{(b)} \cdot \frac{1}{K_b}$
3. Assemble the block score vector: $\mathbf{t}_{a,\text{new}}^{[s]} = \left[t_{a,\text{new}}^{(1)}, \ \ldots \ , t_{a,\text{new}}^{(B)} \right]$
4. Calculate the super score: $t_{a,\text{new}}^{(s)} = \mathbf{t}_{a,\text{new}}^{[s]}\mathbf{w}_a^{[s]}$
5. Deflate each block: $\mathbf{x}_{\text{new}}^{(b)} = \mathbf{x}_{\text{new}}^{(b)} - t_{a,\text{new}}^{(s)}\mathbf{p}_a^{(b)}$ using *superscore*
6. Repeat from step 2 for all components $a = 1, 2, \ldots A$
7. Predict: $\hat{\mathbf{y}}_{\text{new}} = t_{1,\text{new}}^{(s)}\mathbf{c}_1 + \ \ldots \ + t_{A,\text{new}}^{(s)}\mathbf{c}_A$

Also calculate SPE and T^2 for each block, and for the super level

Which deflation to use for MBPLS

Method 1

- Removes all variation in $\mathbf{t}_a^{(b)}$ from $\mathbf{X}^{(b)}$
- Also, $\mathbf{t}_a^{(b)} w_{a,b}^{[s]}$ is the portion from block b used to predict $\mathbf{Y}$
- If $\mathbf{w}_{a,b}^{[s]} \approx 0$ (small super weight for block b for component a), then $\mathbf{t}_a^{(b)}$ has not predictive ability for $\mathbf{Y}$
- Once removed (deflated), it cannot be used in subsequent components
- One advantage though: the block scores tend to be more directly related to $\mathbf{Y}$

Method 2

- Removes from $\mathbf{X}^{(b)}$ the variation in $\mathbf{t}_a^{(s)}$
- Variation in $\mathbf{t}_a^{(s)}$ is used to explain $\mathbf{Y}$

Actual calculation for MBPLS

Westerhuis, Kourti and MacGregor (1998) showed we don't need to calculate MBPLS as just described.

Easier approach:

- Preprocess the data from each block as normal
- Post divide each block by $\sqrt{K_b}$ and assemble:

$$\mathbf{X} = \left[\frac{\mathbf{X}^{(1)}}{\sqrt{K_1}},\ \frac{\mathbf{X}^{(2)}}{\sqrt{K_2}},\ \ldots,\ \frac{\mathbf{X}^{(B)}}{\sqrt{K_B}} \right]$$

- Calculate PLS in the usual way on **X** and **Y** to obtain:
  - scores are identical to MBPLS super scores, $\mathbf{t}_1^{(s)}, \mathbf{t}_2^{(s)}, \ldots, \mathbf{t}_A^{(s)}$
  - back-calculate the block weights, loadings and scores
  - then calculate the block SPE and T^2
  - also calculate the super weights
  - results will be identical to the full approach

Is all this complexity worth it?

Given the above derivations (especially if this is the first time seeing it), one can rightly ask whether this is worth it.

- Consensus PCA can be calculated from ordinary PCA
- Multiblock PLS can be calculated from ordinary PLS
- This implies the predictive performance will be identical

Advantages are:

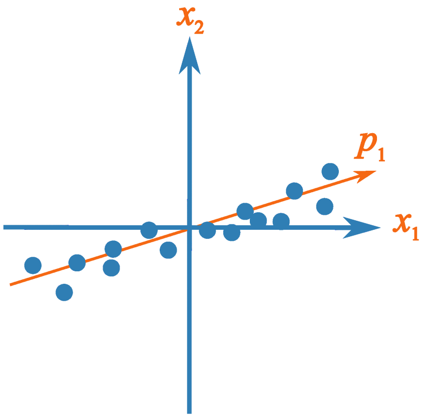
- better interpretation
- separate monitoring and fault detection for each block, since
  - each block has its own SPE, T^2, weights, loadings, VIP, R^2
  - super level: has SPE, T^2, weights, VIP, R^2

Centering

- Removes arbitrary offset from each column
- Not centering: sometimes results in extra component
- If centering: sometimes get a better fit
- Centering: can have a major impact on model's interpretation
- May be done around a "natural" reference: e.g. setpoint, instead of the mean
- For robustness: center about the median instead

More details from this very insightful paper: Bro and Smilde: "Centering and scaling in component analysis"

Scaling

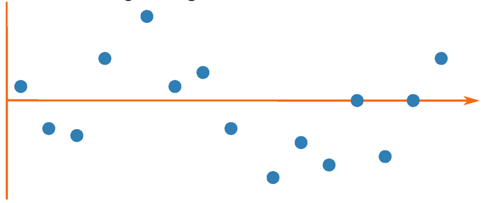


- If no prior knowledge: scale each column to unit variance
- Need to emphasize/deemphasize a variable?
  - First scale all columns to unit variance
  - Then upscale/downscale the column(s) as appropriate

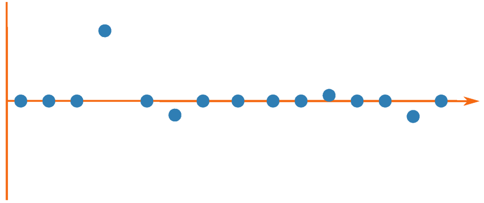
Scaling

Be careful of inflating noise in variables with little signal

Variable with good signal



Variable with little signal (low standard deviation)

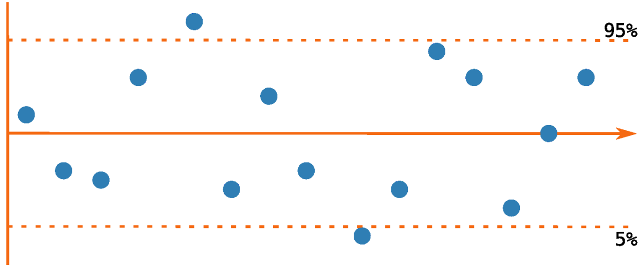


- Use robust scaling instead (scale by MAD)
- Heuristic: don't scale x_k if $\text{stdev}(x_k) < 4 \times$ its measurement noise

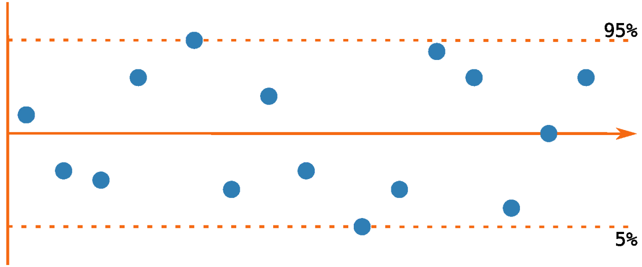
$$\text{median absolute deviation}_k = 1.4826 \cdot \text{median}\left\{ \left| x_k - \text{median}\left\{x_k\right\} \right| \right\}$$

Dealing with outliers: winsorizing

Raw data



Winsorized data

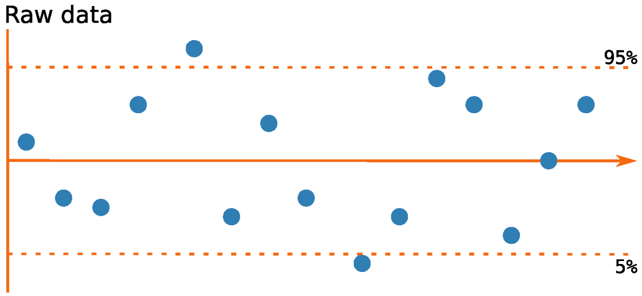


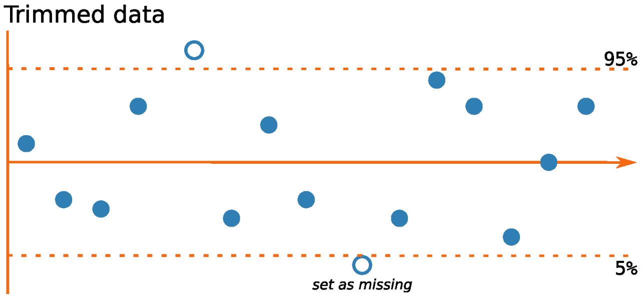
Replace outliers beyond a chosen α level at their respective bounds

e.g. $\alpha = 5\%$ shown here

Can break multivariate relationships between variables

Dealing with outliers: trimming





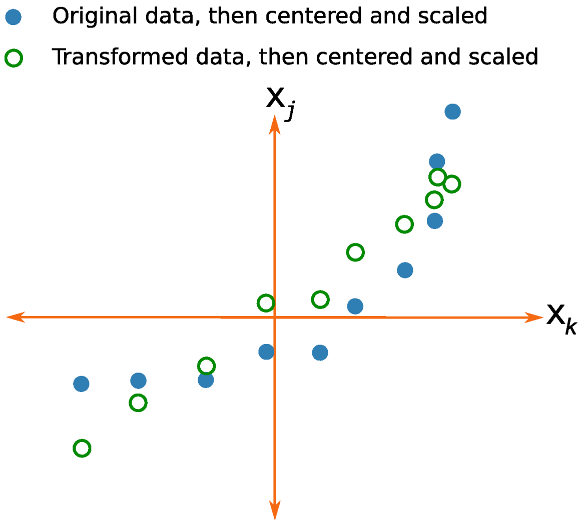
Convert outliers beyond a chosen α level to missing values

e.g. $\alpha = 5\%$ shown here

Better (more honest) alternative to winsorizing

Univariate transformations

Univariate transformation's often motivated by first principle's knowledge:



Examples:

- Use $\log(T)$ instead of temperature, T
- Use $1/P$ instead of pressure, P
- $\sqrt{F}$ instead of flow, F (as shown in figure)

Univariate transformations

- Pre-bias data to avoid negative and zero values
- e.g. if T is measured in Celcius and could be negative, use $\log(T + c)$
- where c is large enough to avoid negative logs

- Can be used on **X and Y** variables
  - e.g. use $\log(y)$ to get better predictions: software takes care of transforming and un-transforming

Expanding the **X**-matrix with calculated variables

Add extra columns to **X**:

- Heat balance:
  - add column for sum of all heat inputs
  - add column for sum of heat outputs
  - add column for sum of heat created/lost by reaction
- As above, but for mass balance
  - See "Data reconciliation" (Tong and Crowe's work)
- Key performance indicators for your particular industry
- Add dimensionless numbers:
  - Reynolds number $= \dfrac{\rho v D}{\mu}$
  - Power number $= \dfrac{P}{\rho n^3 d^5}$

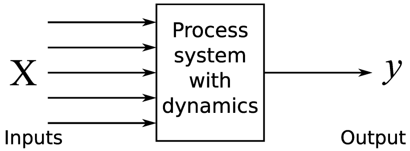
Soft-sensor applications:

- Add columns from first-principles equations, e.g. Antoine equation: $\log(\text{RVP}) = \log P - B\left[\dfrac{1}{C} - \dfrac{1}{T+C}\right]$

Handling process dynamics with lagging

Time series modelling is a well-studied topic*

- Inputs: $\mathbf{x}(t)$
- Outputs: $\mathbf{y}(t)$



- Time series models can be built to predict $\mathbf{y}_t$ when given:
  - $\mathbf{x}_t, \mathbf{x}_{t-1}, \mathbf{x}_{t-2}, \ldots$
  - $\mathbf{y}_{t-1}, \mathbf{y}_{t-2}, \ldots$
- These models are then used for:
  - forecasting: to make decisions
  - control: e.g. feedback control
  - process monitoring

- Can we include these ideas in PCA/PLS?

* Standard reference: Box and Jenkins; book by Chatfield is very good

Time series example

- First order system's transfer function: $\dfrac{y(s)}{x(s)} = \dfrac{K}{\tau s + 1}$
- Time domain: $\tau \dfrac{dy}{dt} + y(t) = Kx(t)$
- Take samples Δt time units apart:

$$y_t = \delta y_{t-1} + K(1-\delta)x_t \quad \text{where} \quad \delta = e^{-\frac{\Delta t}{\tau}}$$

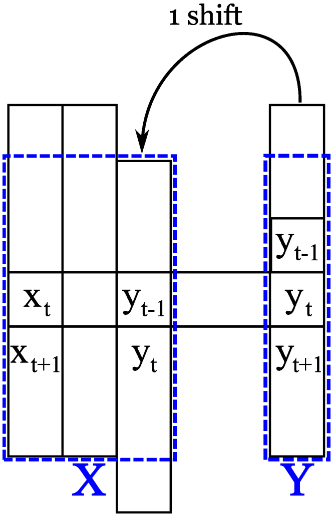
General form:

$$y_t = a_1 y_{t-1} + a_2 y_{t-1} + \ldots + a_m y_{t-m} + b_0 x_t + b_1 x_{t-1} + \ldots + b_n x_{t-n}$$

- function of past y's: $y_{t-1}, y_{t-2}, \ldots$
- function of current and past x's: $x_t, x_{t-1}, \ldots$

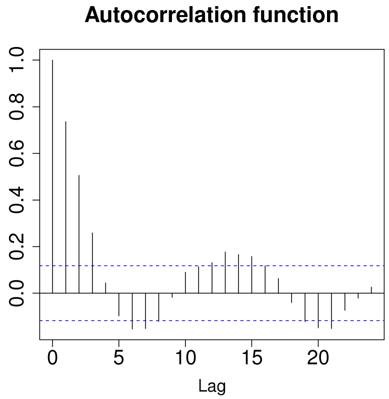
Time series example: lagging Y's

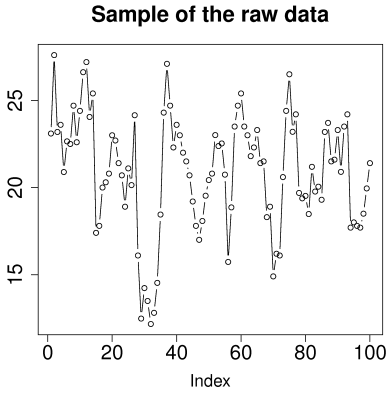
- How to approximate this with a latent variable model?
- Copy and shift columns to get consistency within a row
- So add columns to **X** to get the time-history of y
- $y(t) = a_1 y(t-1) + b_0 x_t + \dots$



Time series example: lagging Y's

- Lagging adds strong correlations among columns in **X**
- That's OK: PLS can handle it.
- What if we don't know how may lags to add?
- One approach: use the autocorrelation function: `acf(y)`

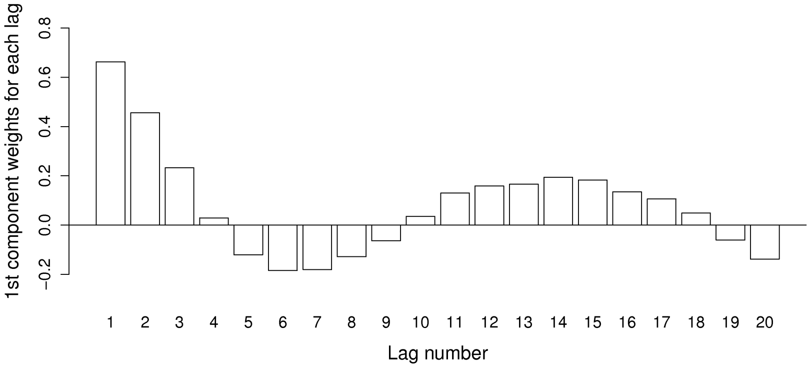




Time series example: lagging Y's

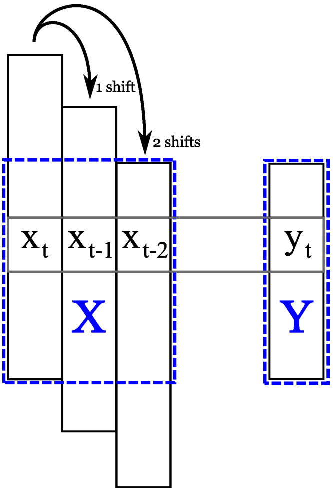
Other approaches:

- look at the PLS coefficients and jack-knifed reliability intervals
- look at the **X**-space weights: $\mathbf{w}^*$

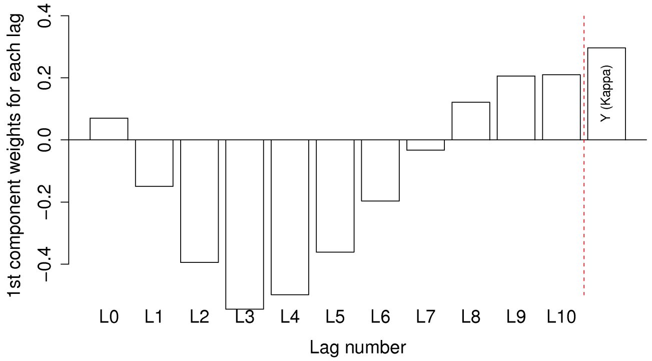


Time series example: lagging X's

- Previous history in **X** variables might be useful
- Especially in processes with long mixing times/residence times
- $y_t = b_0 x_t + b_1 x_{t-1} + b_2 x_{t-2} + \ldots$
- Unsure how many? Add lags then check coefficient and/or $\mathbf{w} * \mathbf{c}$ plots
- Use lags which are large



Lagging



- Sometimes we find counter-intuitive lags
- Sometimes the lags are all small and “smeared” over many columns
  - Then use an average over all important lags instead
  - Better still: block-scale all lags to have equivalent influence of 1 variable

Dealing with nuisance variation

Sometimes we have dominant variation that we are not interested in: e.g. **throughput**

This can dominate a component, and cause false alarms for monitoring. Can't remove throughput variable, because it helps the model. Options?

1. If it is just in one score, e.g. t_2, just skip it in T^2 calculation
2. Regress out the throughput effect:
   - Regress the throughput variable on all other X-variables
   - Take residuals (which are mostly orthogonal now to throughput)
   - Use these residuals as your X
   - Model is hard to interpret though; good for predictions
3. Preprocessing option: divide some X-variables by throughput to normalize out throughput effect; still keep the single throughput variable.