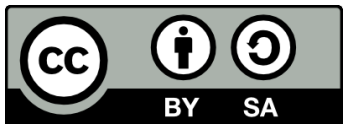
Latent Variable Methods Course

Learning from data

Instructor: Kevin Dunn
kevin.dunn@connectmv.com
http://connectmv.com

Revision: 268:adfd compiled on 15-12-2011

Copyright, sharing, and attribution notice



This license allows you:

- **to share** - to copy, distribute and transmit the work
- **to adapt** - but you must distribute the new result under the same or similar license to this one
- **commercialize** - you *are allowed* to create commercial applications based on this work
- **attribution** - you must attribute the work as follows:
  - "Portions of this work are the copyright of ConnectMV", *or*
  - "This work is the copyright of ConnectMV"

We appreciate:

- if you let us know about **any errors** in the slides
- **any suggestions to improve the notes**
- telling us if you use the slides, especially commercially, so we can inform you of major updates
- emailing us to ask about different licensing terms

All of the above can be done by writing us at

`courses@connectmv.com`

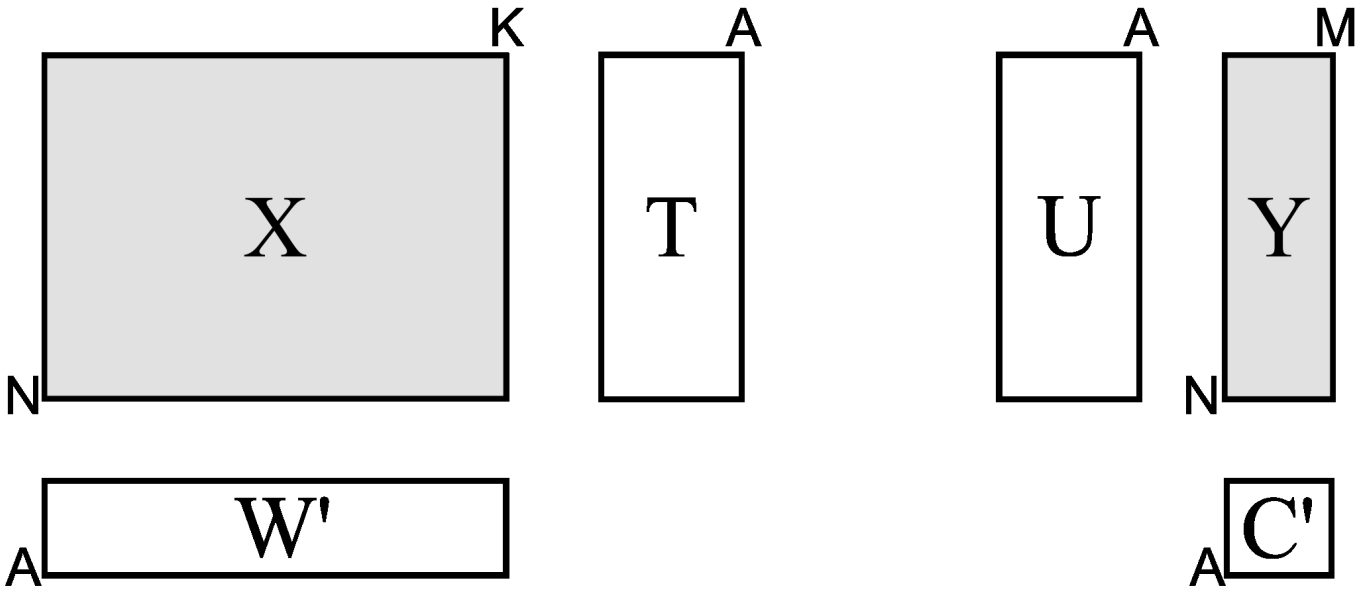
If reporting errors/updates, please quote the current revision number: `268:adfd`

Overview

Today's class we start with applications. We will look at a main application of PLS: **prediction**

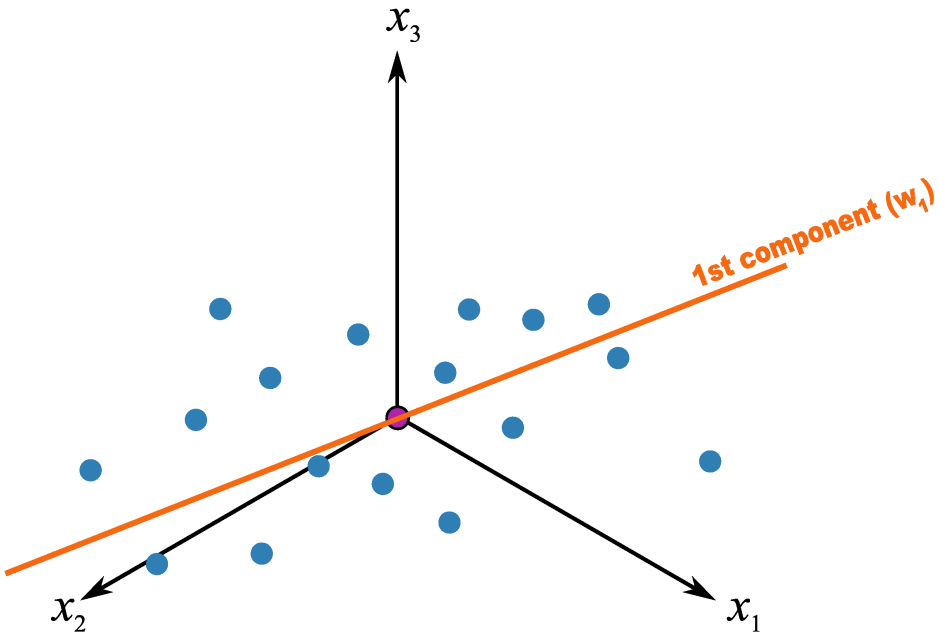
We also learn more about good data analysis practices.

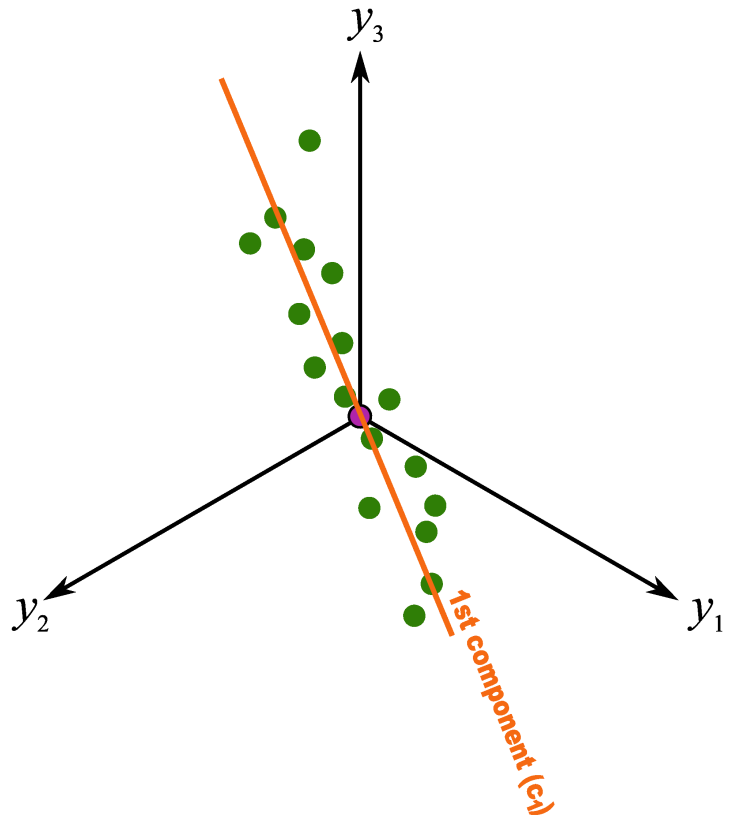
Simple PLS (SIMPLS)



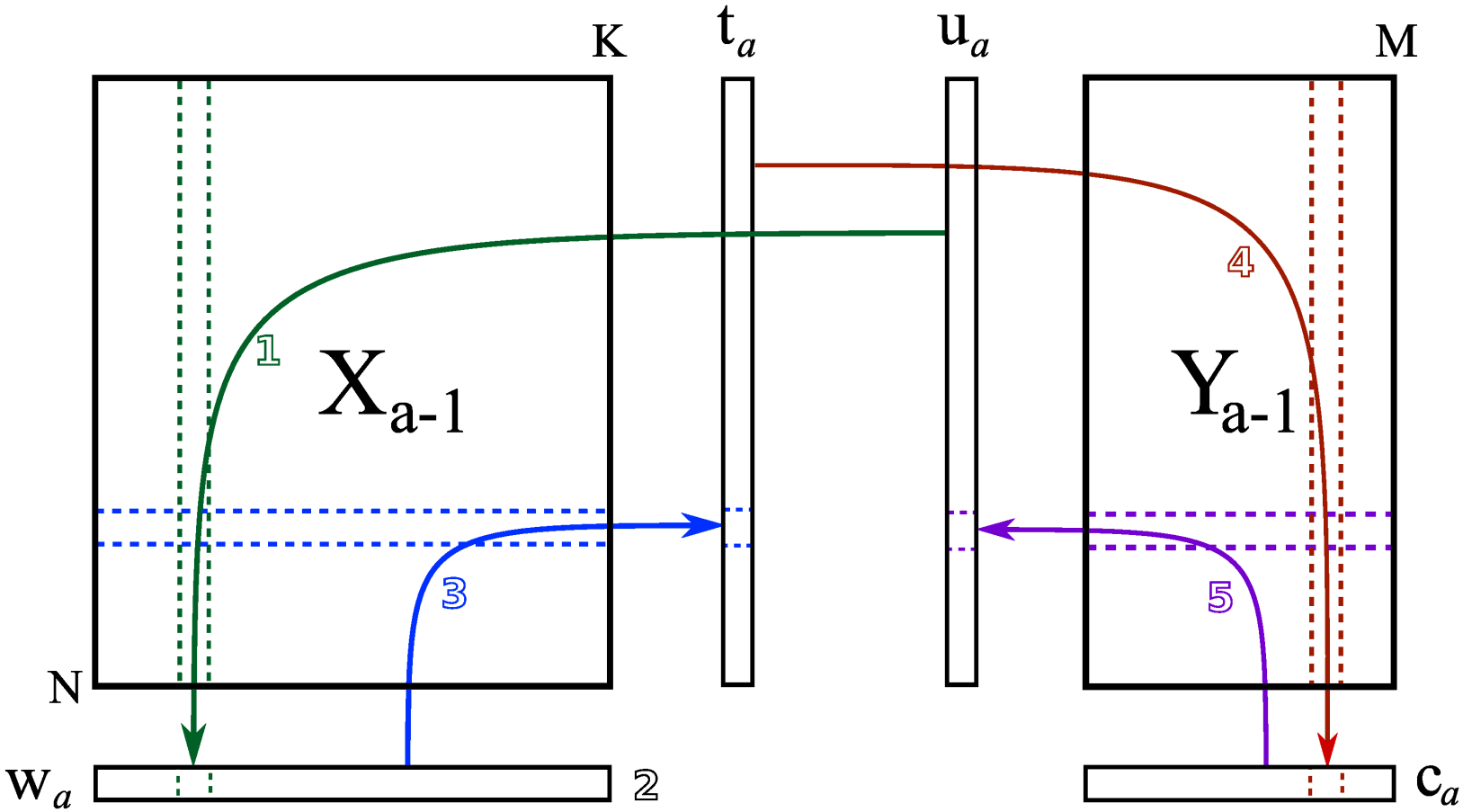
1. PLS scores explain **X**:
   - $\mathbf{t}_a = \mathbf{X}_a \mathbf{w}_a$ for the **X**-space
   - max : $\mathbf{t}_a' \mathbf{t}_a$ subject to $\mathbf{w}_a' \mathbf{w}_a = 1.0$
2. PLS scores also explain **Y**:
   - $\mathbf{u}_a = \mathbf{Y}_a \mathbf{c}_a$ for the **Y**-space
   - max : $\mathbf{u}_a' \mathbf{u}_a$ subject to $\mathbf{c}_a' \mathbf{c}_a = 1.0$
3. PLS maximizes relationship between **X**- and **Y**-space
   - maximizes covariance: $\mathrm{Cov}(\mathbf{t}_a, \mathbf{u}_a)$
   - $\mathrm{Cov}(\mathbf{t}_a, \mathbf{u}_a) = \mathrm{Corr}(\mathbf{t}_a, \mathbf{u}_a) \cdot \sqrt{\mathbf{t}_a' \mathbf{t}_a} \cdot \sqrt{\mathbf{u}_a' \mathbf{u}_a} \cdot \frac{1}{N}$

PLS: geometric interpretation





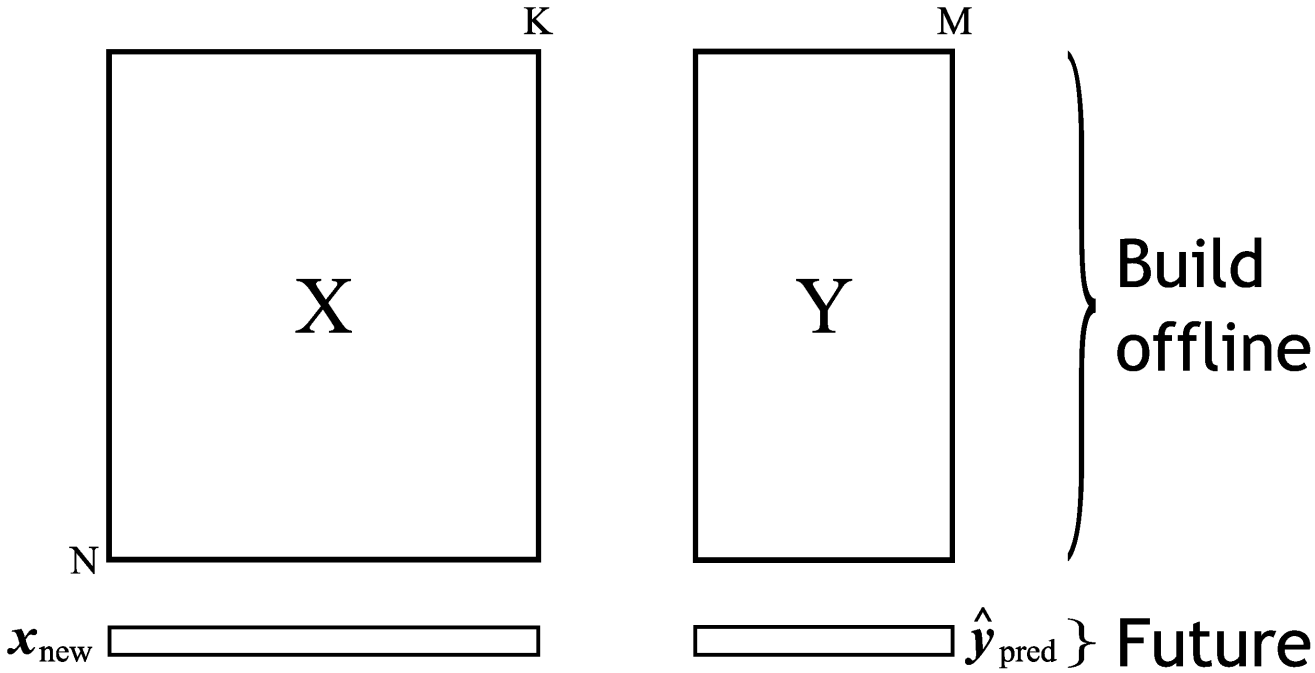
NIPALS algorithm for PLS



The weights in PLS

- Scores are calculated from deflated matrices:
  - $\mathbf{t}_1 = \mathbf{X}_{a=0}\mathbf{w}_1 = \mathbf{X}_0\mathbf{w}_1$
  - $\mathbf{t}_2 = \mathbf{X}_{a=1}\mathbf{w}_2 = (\mathbf{X}_0 - \mathbf{t}_1\mathbf{p}_1)\,\mathbf{w}_2$
- $\mathbf{w}_2$: relates score $\mathbf{t}_2$ to $\mathbf{X}_{a=1}$, the deflated matrix
- This is hard to interpret. We would like instead:
  - $\mathbf{t}_1 = \mathbf{X}_{a=0}\mathbf{w}*_1 = \mathbf{X}_0\mathbf{w}*_1$
  - $\mathbf{t}_2 = \mathbf{X}_{a=0}\mathbf{w}*_2 = \mathbf{X}_0\mathbf{w}*_2$
  - *etc*
- We calculate matrix $\mathbf{W}^* = \mathbf{W}\,(\mathbf{P}'\mathbf{W})^{-1}$
- So $\mathbf{T} = \mathbf{X}_0\mathbf{W}^*$, or simply: $\boxed{\mathbf{T} = \mathbf{X}\mathbf{W}^*}$
  - $\mathbf{w}*_1 = \mathbf{w}_1$
  - $\mathbf{w}*_a \neq \mathbf{w}_a$ for $a > 1$
- We get a clearer interpretation of the variable relationships using $\mathbf{W}^*$ instead of $\mathbf{W}$

Using PLS on new data

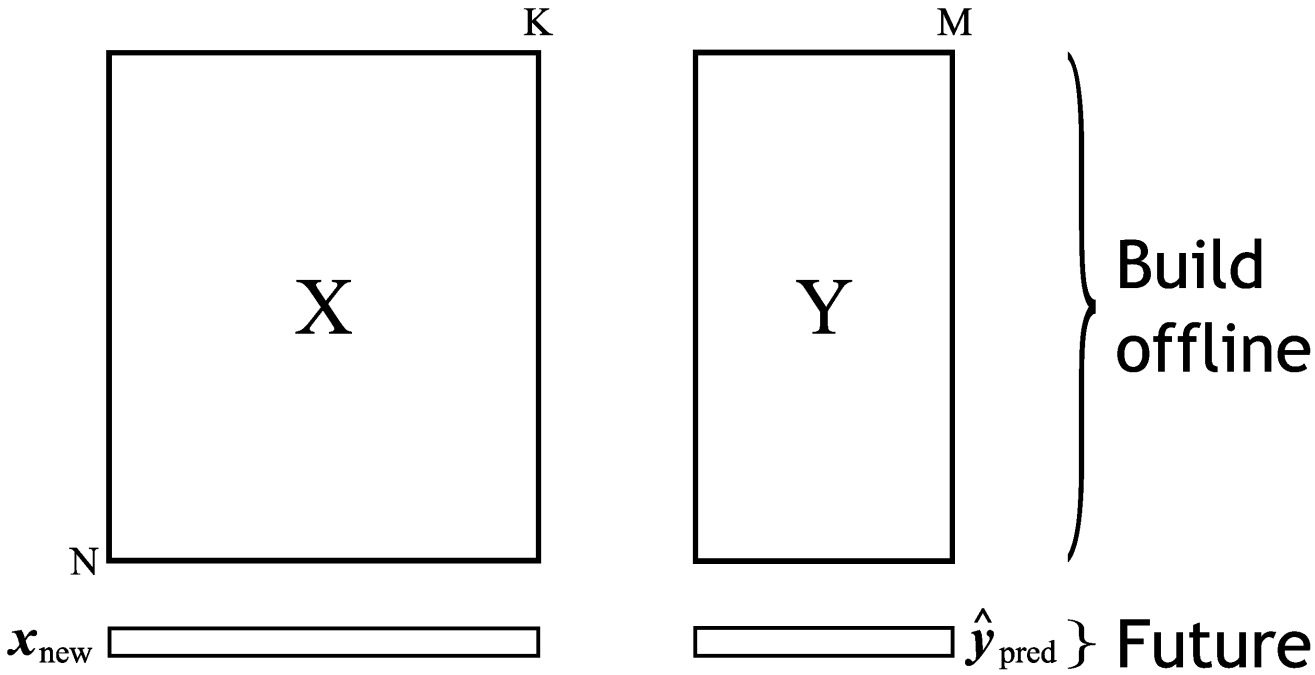


$$
\begin{aligned}
t_{1,\text{new}} &= \mathbf{x}'_{\text{new}}\mathbf{w}_1 \\
\mathbf{x}'_{\text{new}} &= \mathbf{x}'_{\text{new}} - t_{1,\text{new}}\mathbf{p}'_1 \qquad \text{(deflate)} \\
t_{2,\text{new}} &= \mathbf{x}'_{\text{new}}\mathbf{w}_2 \\
\mathbf{x}'_{\text{new}} &= \mathbf{x}'_{\text{new}} - t_{2,\text{new}}\mathbf{p}'_2 \\
&\textit{etc}
\end{aligned}
$$

Collect all the $t_{\text{a,new}}$ score values in $\mathbf{t}_{\text{new}}$

Alternatively use $\mathbf{t}_{\text{new}} = \mathbf{x}'_{\text{new}}\mathbf{W}^*$ to get $\mathbf{t}_{\text{new}}$ without deflation

Using PLS on new data



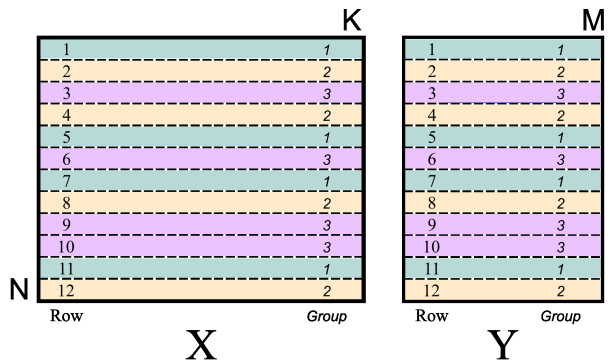
$$\begin{aligned} \widehat{\mathbf{y}}'_{new} &= \mathbf{t}'_{new}\mathbf{C}' \\ \widehat{\mathbf{y}}'_{new} &= \mathbf{x}'_{new}\mathbf{W}^*\mathbf{C}' \end{aligned}$$

- Then uncenter and unscale the $\widehat{\mathbf{y}}'_{new}$

Cross-validation to calculate Q^2

Similar procedure as with PCA

Split the rows in **X** and **Y** into G groups.

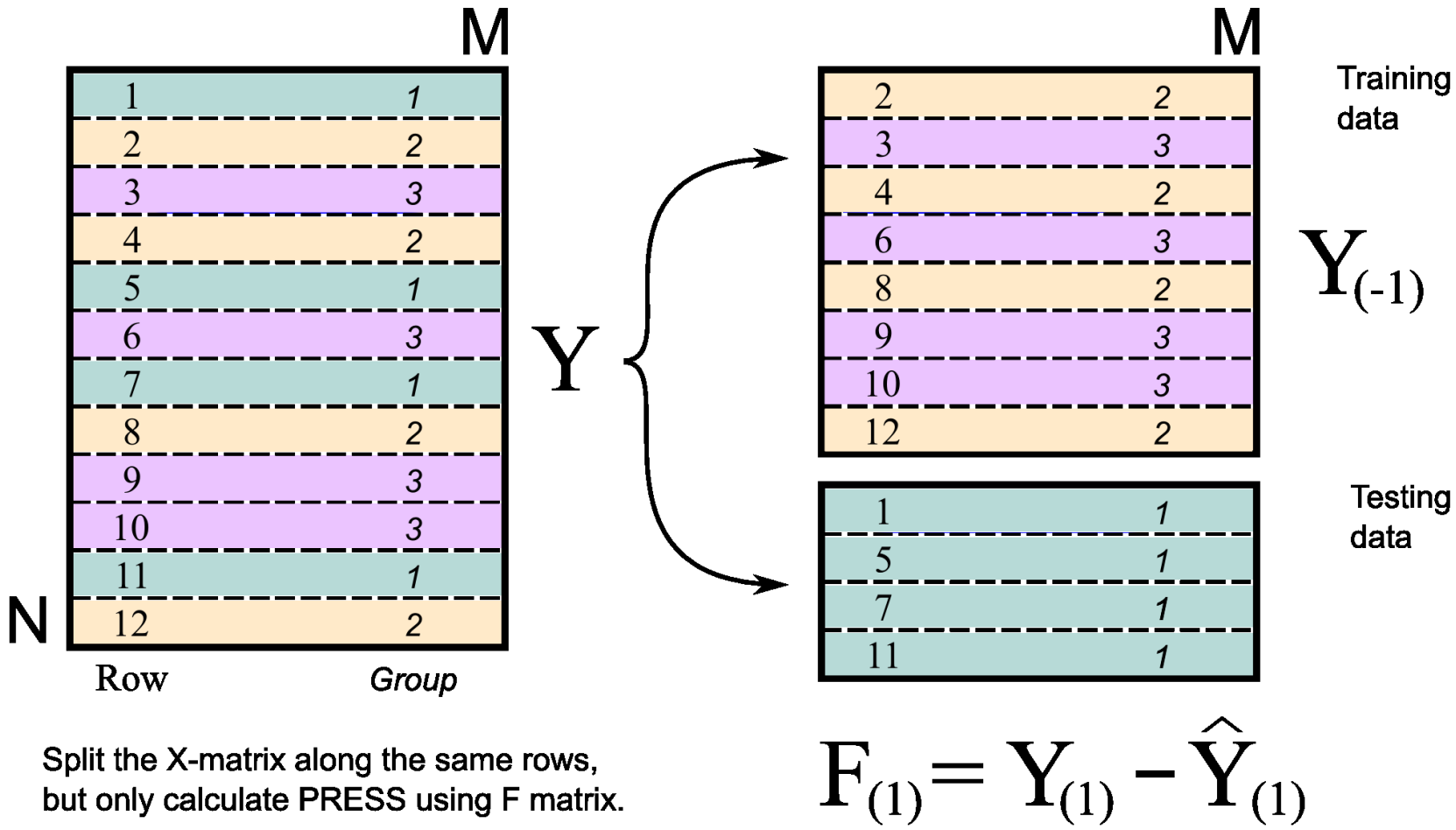


$G = 3$ in this illustration

- Typically $G \approx 7$ [ProSensus, Simca-P use $G = 7$]
- Rows can be randomly grouped, or
- ordered *e.g.* 1, 2, 3, 1, 2, 3, ...
- ordered *e.g.* 1, 1, 2, 2, 3, 3, ...

Cross-validation concept for PLS

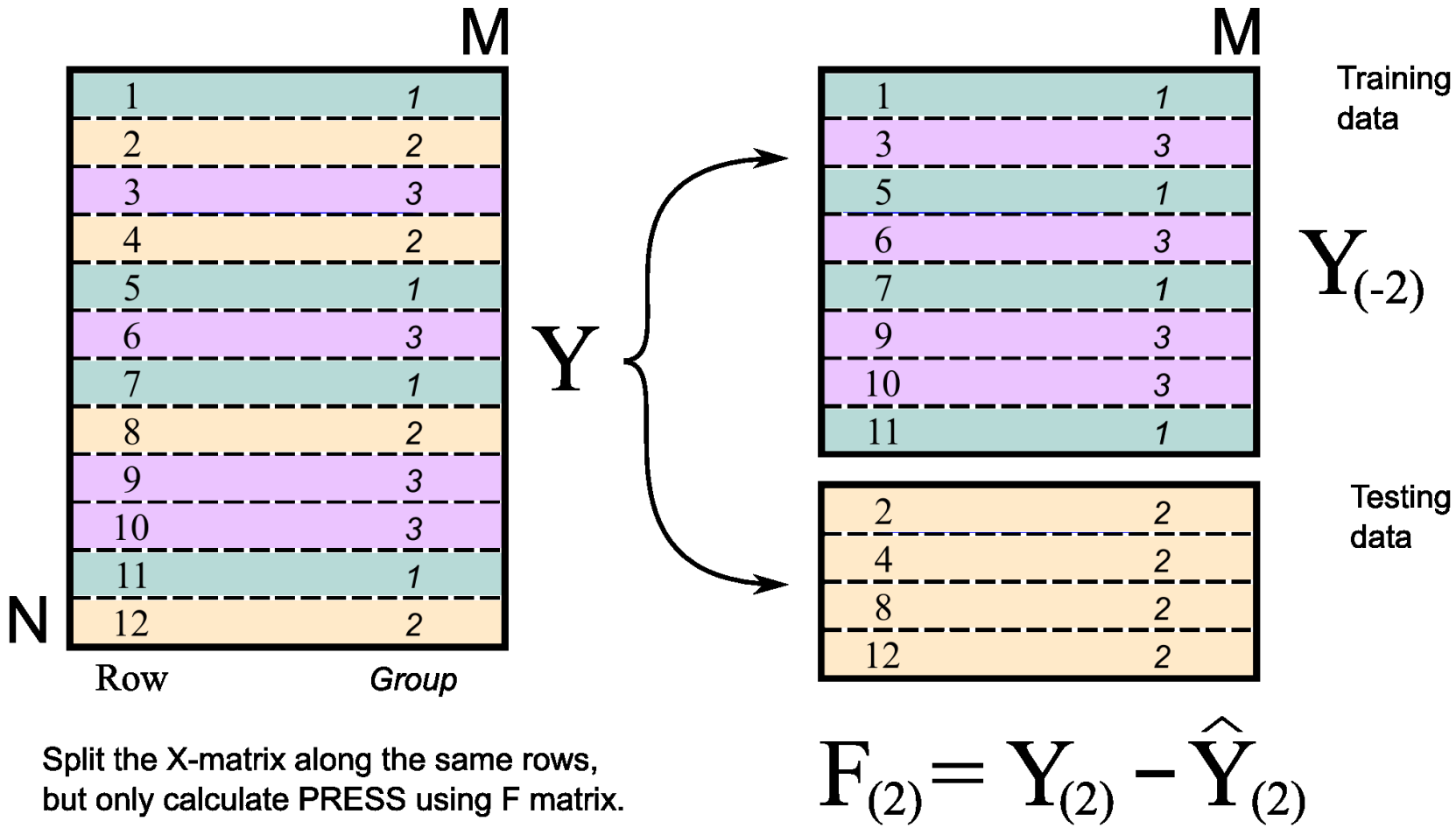
Fit a PLS model using $\mathbf{X}_{(-1)}$ and $\mathbf{Y}_{(-1)}$; use $\mathbf{X}_{(1)}$ as testing data



$\mathbf{F}_{(1)}$ = prediction error for testing group 1

Cross-validation concept for PLS

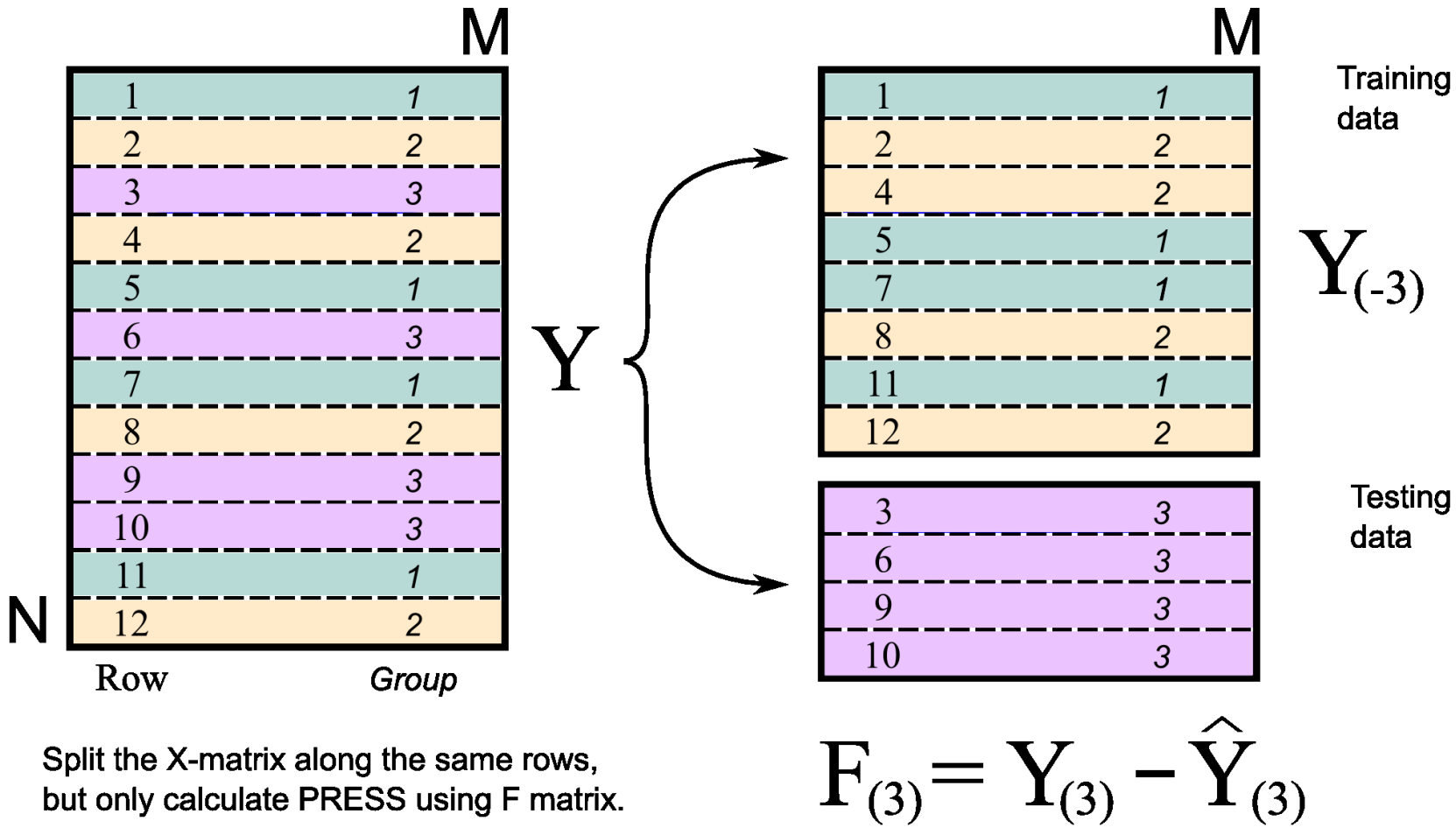
Fit a PLS model using $\mathbf{X}_{(-2)}$ and $\mathbf{Y}_{(-2)}$; use $\mathbf{X}_{(2)}$ as testing data



$\mathbf{F}_{(2)}$ = prediction error for testing group 2

Cross-validation concept for PLS

Fit a PLS model using $\mathbf{X}_{(-3)}$ and $\mathbf{Y}_{(-3)}$; use $\mathbf{X}_{(3)}$ as testing data



$\mathbf{F}_{(3)}$ = prediction error for testing group 3

Cross-validation concept for PLS

- PRESS $= \text{ssq}(\mathbf{F}_{(1)}) + \text{ssq}(\mathbf{F}_{(2)}) + \ldots + \text{ssq}(\mathbf{F}_{(G)})$
- PRESS = prediction error sum of squares from each prediction group
- $Q^2 = 1 - \dfrac{\mathcal{V}(\text{predicted } \mathbf{F}_A)}{\mathcal{V}(\mathbf{Y})} = 1 - \dfrac{\text{PRESS}}{\mathcal{V}(\mathbf{Y})}$
- Q^2 is calculated and interpreted in the same way as R^2
- Q^2_k can be calculated for variable $k = 1, 2, \ldots K$
- You should always find $Q^2 \leq R^2$
- If $Q^2 \approx R^2$: that component is useful and predictive in the model
- If Q^2 is "small": that component is likely fitting noise

To read: Esbensen and Geladi, 2010, "Principles of proper validation"

PLS plots

- Score plots: **t** and **u** show relationship between rows
- Weight plots: **w**: relationship between **X** columns
- Loading plots: **c**: relationship between **Y** variables
- Weight and loading plots: $\mathbf{w}^*\mathbf{c}$: relationship between **X** and **Y**
- SPE plots (X-space, Y-space)
- Hotelling's T^2 plot
- Coefficient plots
- VIP plot
- R^2 plots (X-space, Y-space, per variable)

Variable importance to prediction

Important variables in the model?

- Have large (absolute) weights: why?
- Come from a component that has a high R^2

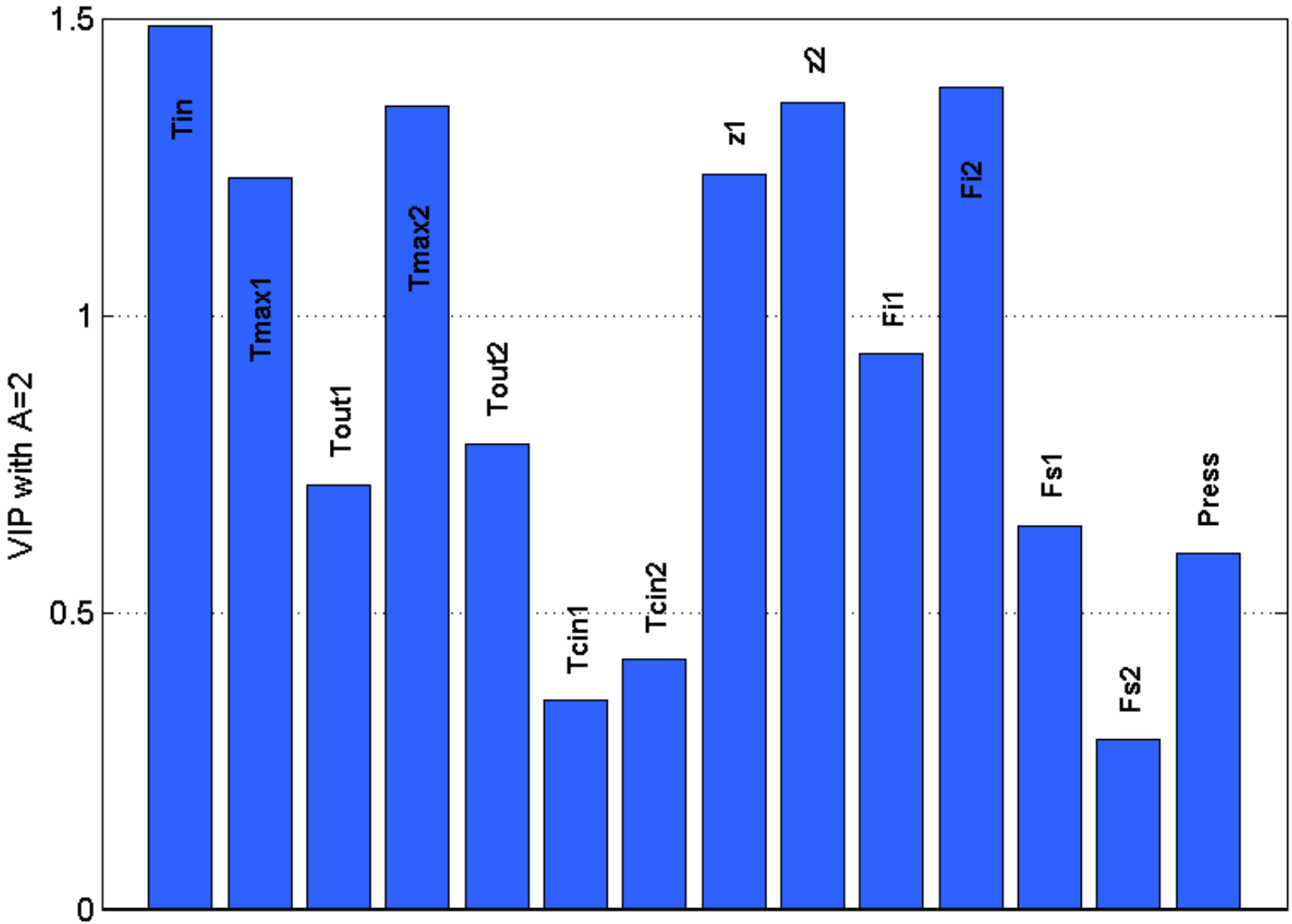
Combining these two concepts we calculate *for each variable*:

Importance of variable k using A components in PLS

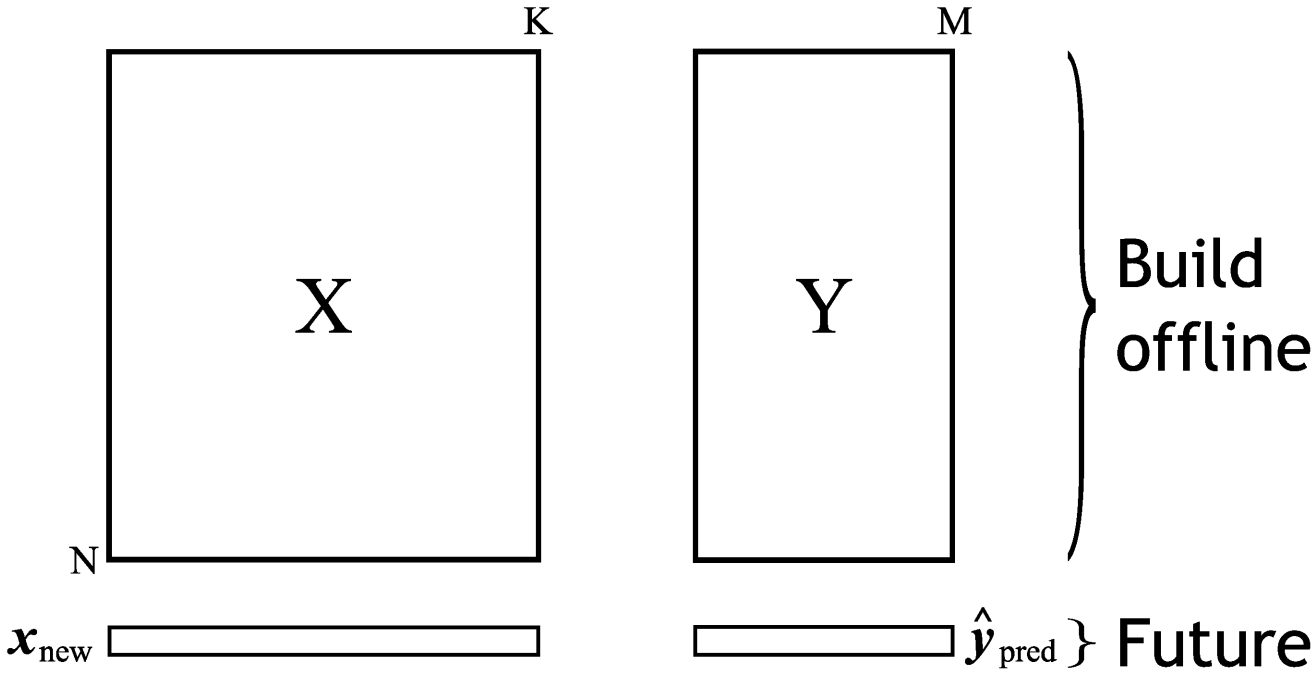
$$VIP^2_{A,k} = \frac{K}{\mathrm{SSX}_0 - \mathrm{SSX}_A} \cdot \sum_{a=1}^{A} (\mathrm{SSX}_{a-1} - \mathrm{SSX}_a)\, W^2_{a,k}$$

- SSX_a = sum of squares in the $\mathbf{X}$ matrix after a components
- $\frac{\mathrm{SSX}_{a-1} - \mathrm{SSX}_a}{SSX_A}$ = incremental R^2 for a^{th} component
- $\frac{\mathrm{SSX}_0 - \mathrm{SSX}_A}{SSX_A} = R^2$ for model using A components
- Messy, but you can show that $\sum_k VIP^2_{A,k} = K$
- Reasonable cut-off $= 1$
- VIP for PCA models: use $P^2_{a,k}$ instead of $W^2_{a,k}$

Variable importance to prediction



Coefficient plot

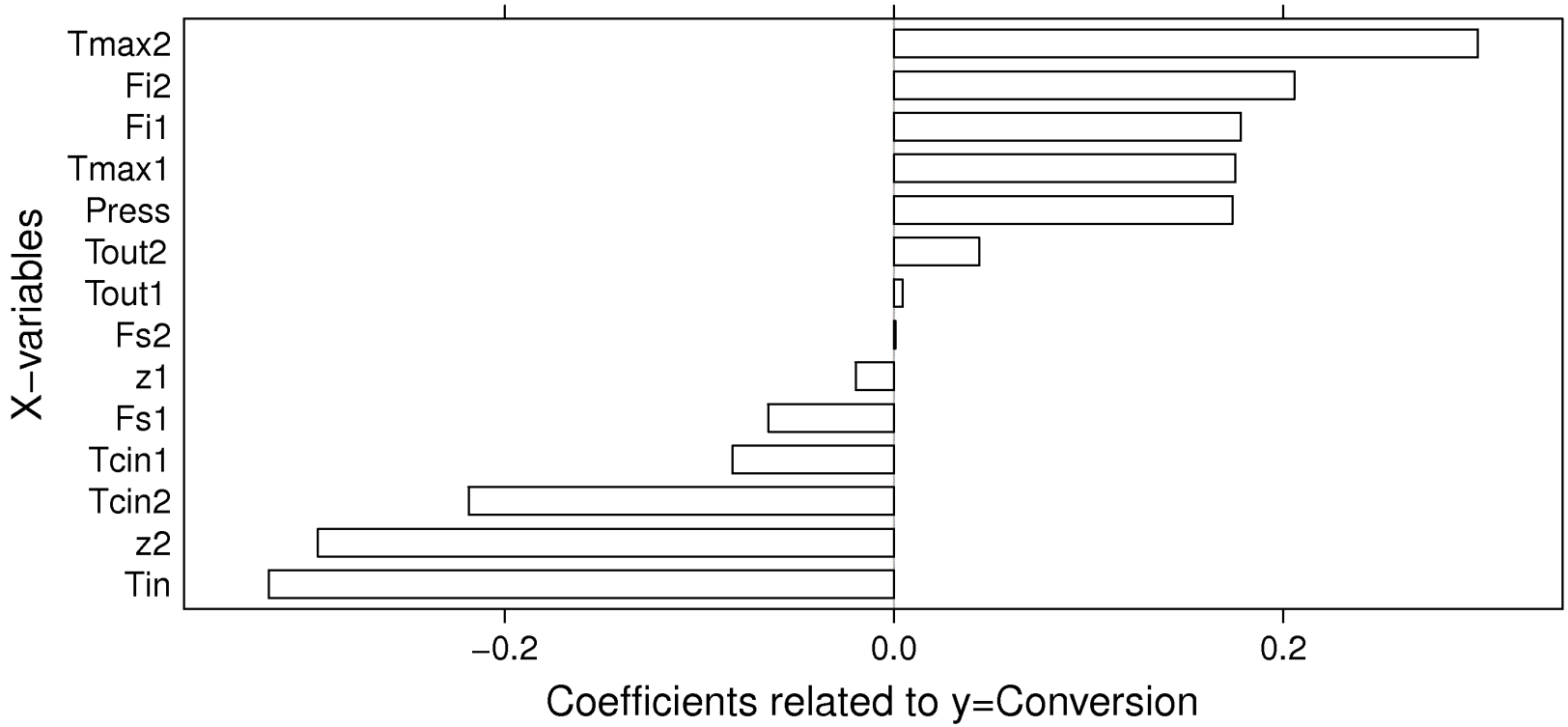


$$\begin{aligned}
\widehat{\mathbf{y}}'_{new} &= \mathbf{t}'_{new}\mathbf{C}' \\
\widehat{\mathbf{y}}'_{new} &= \mathbf{x}'_{new}\mathbf{W}^*\mathbf{C}' \\
\widehat{\mathbf{y}}'_{new} &= \mathbf{x}'_{new}\beta
\end{aligned}$$

- β is a $K \times M$ matrix
- Each column in β contains the regression coefficients for column m from $\mathbf{Y}$ matrix
- **Never implement PLS using β matrix**

Coefficient plot

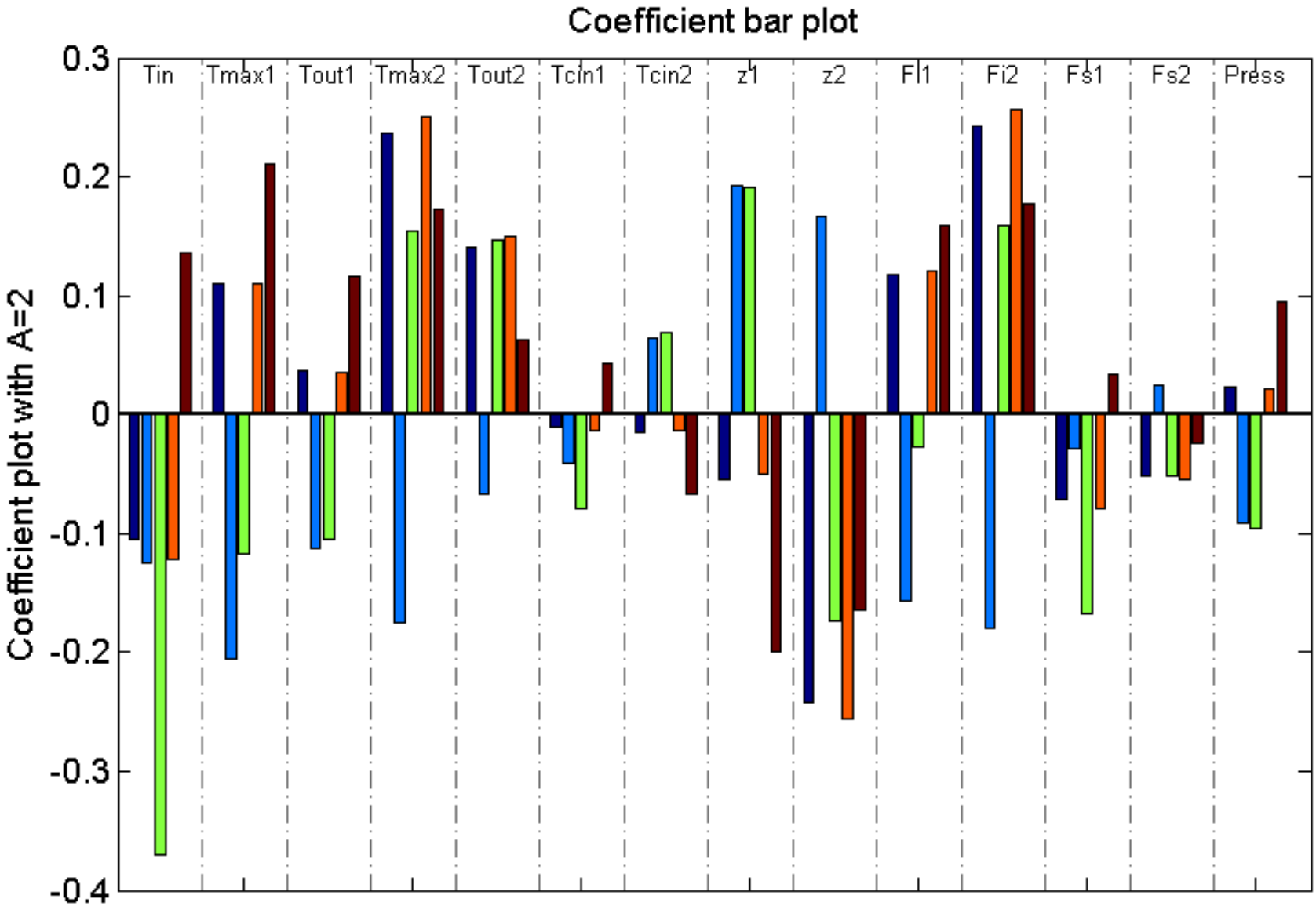
For a single y-variable:



- $\hat{y} = \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_K x_K$
- where x_k and $\hat{y}$ are the preprocessed values
- *Again* – never implement PLS this way.

Coefficient plot

For all **Y**-variables



Jackknifing

We re-calculate the model $G + 1$ times during cross-validation:

- G times, once per group
- The "+1" is from the final round, where we use **all** observations

We get $G + 1$ estimates of the model parameters:

- loadings
- VIP values
- coefficients

for every variable $(1, 2, \ldots K)$.

Calculate "reliability intervals" (don't call them confidence intervals)

- Martens and Martens (paper 43) describe jackknifing.
- Efron and Tibshirani describe the bootstrap and jackknife.

Soft sensors

The problem

Certain measurements/final properties [called y's] are

1. expensive, and/or
2. time-consuming

to obtain from many practical systems.

Aim

Get a prediction of y cheaply and quickly

How can we get predictions?

First-principles models are obviously more desirable.

- based on cause and effect
- can be used to later improve and optimize the process also

But ....

- may be costly to identify the values of parameters in the model
- ... impossible even
- high-fidelity models may take a long time to execute: too slow for real-time use

all of which negate their usefulness.

You might see any number of other models in practice: Kalman filters, neural networks, Bayesian networks, *etc*.

Principle and motivation

Principle

Use **any predictive model** to predict the value of interest, $\hat{y}$, from any other available data, $\mathbf{X}$

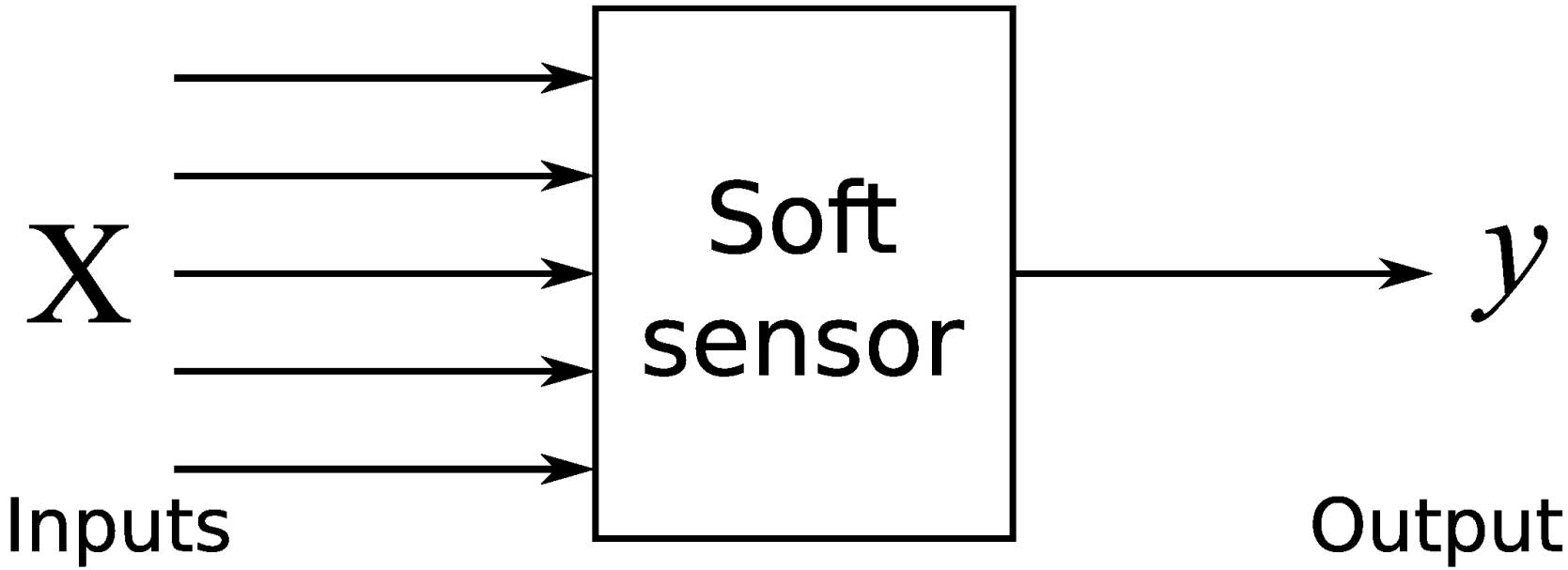
This predictive model is called:

- a soft sensor, or
- an inferential sensor

These predictions are mainly used for:

- monitoring of that $\hat{y}$
- feedback control to maintain $\hat{y}$ at target

Soft sensor structure



Value of interest, y, should ideally be measured on the process

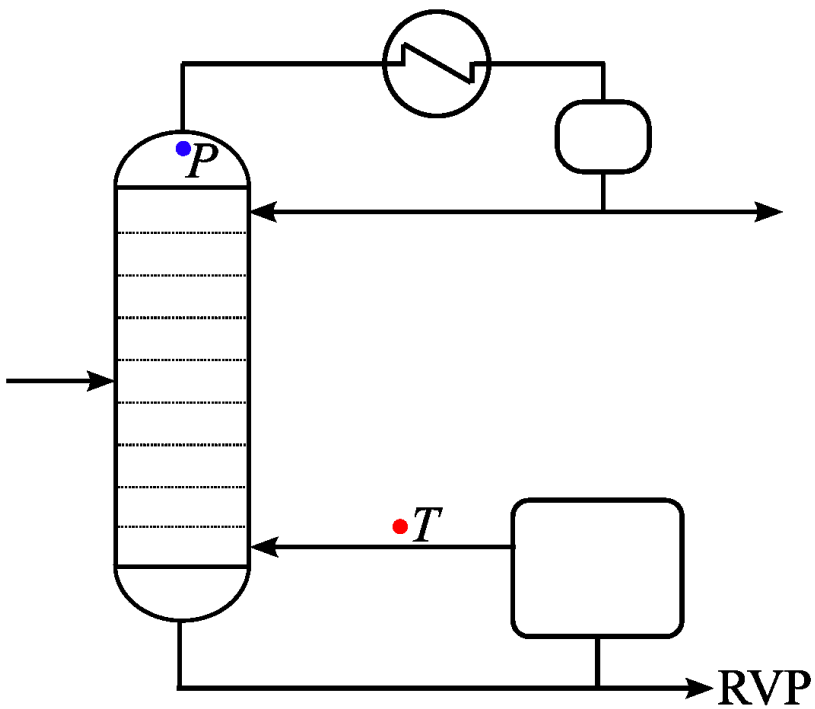
Data used in $\mathbf{X}$:

- real-time process measurements
- calculated variables from the real-time measurements
- spectra, acoustical data, image data
- categorical variables

... anything that is readily available as input into the model to predict y

Example: distillation column purity prediction

- Predict purity from distillation column bottom $= y =$ RVP
- Inputs to model:
  - Various temperatures on the column
  - Pressure measurements
  - Flow rates through the column (column demand)
  - External factors: ambient temperature



Example: distillation column purity prediction

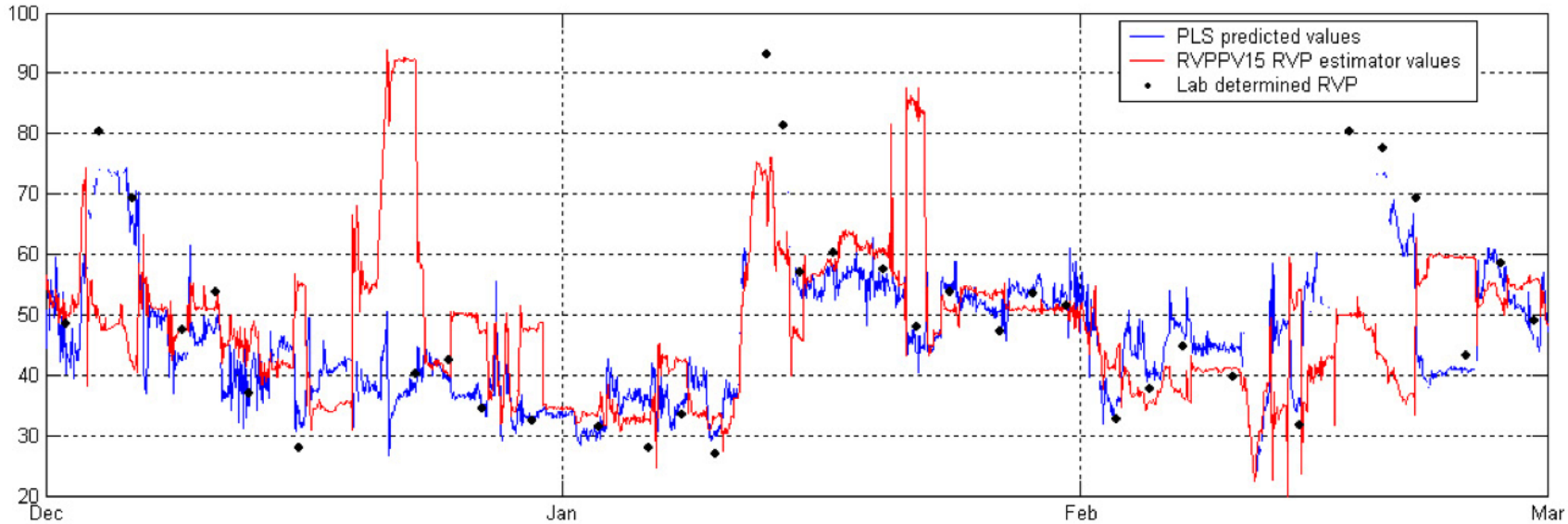
We know the Antoine equation applies here:

$$\log(\text{RVP}) = \log P - B\left[\frac{1}{C} - \frac{1}{T+C}\right]$$

Actual use: $\log(\text{RVP}) = \log P - B\left[\frac{1}{C} - \frac{1}{T+C}\right] + \text{bias}$

Bias is updated 3 times per week to "correct" predictions (accuracy)

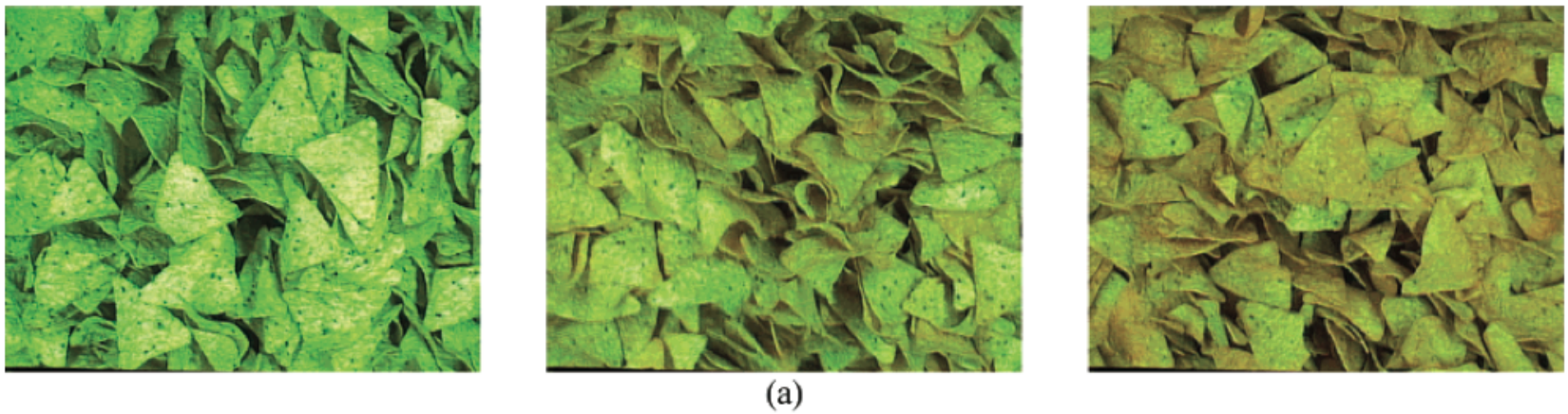
Example: distillation column purity prediction



- `RVPPV15 RVP` is the Antoine estimate
- It occasionally gives really poor predictions. *Why*?

Example: snack food seasoning prediction

- Work by Honglu Yu: http://digitalcommons.mcmaster.ca/opendissertations/866/
- Image data used as **X**



Example: snack food seasoning prediction

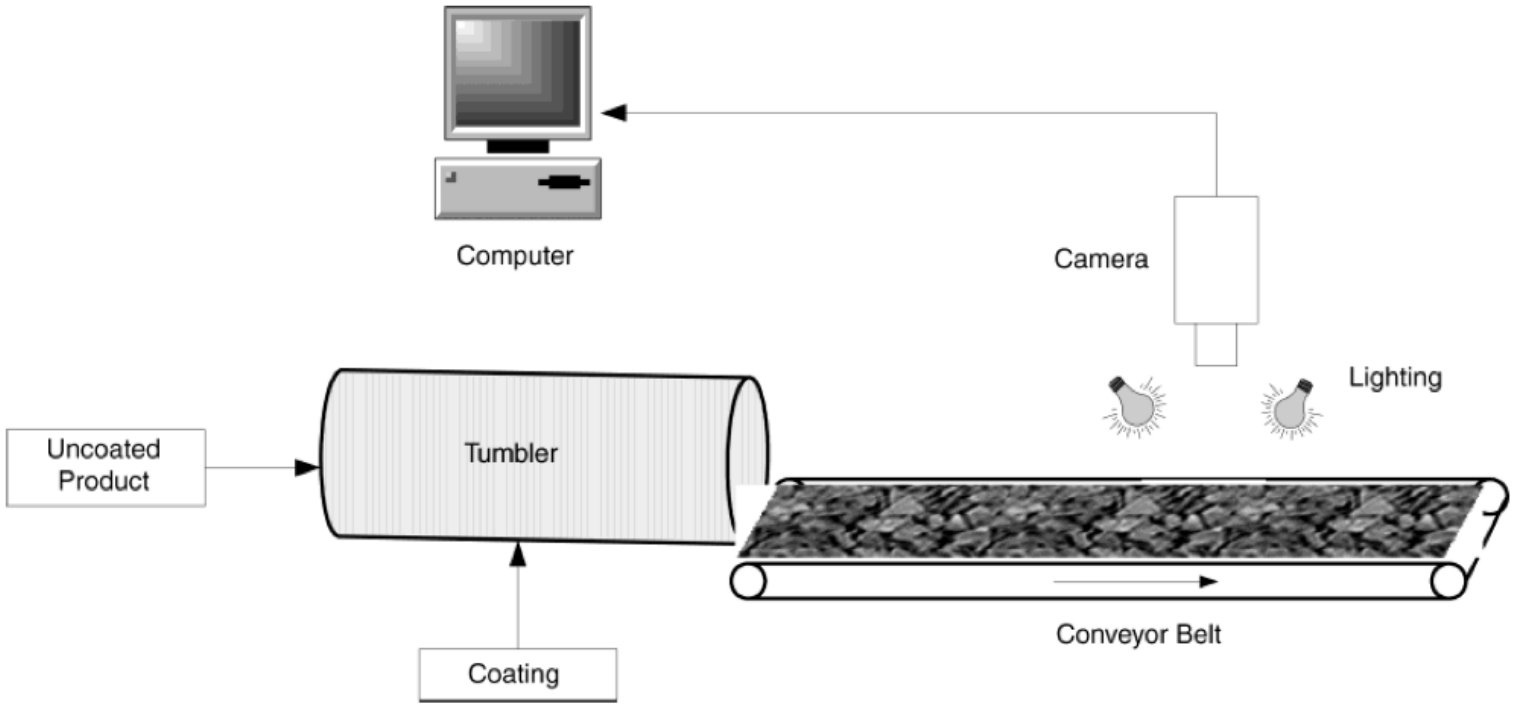
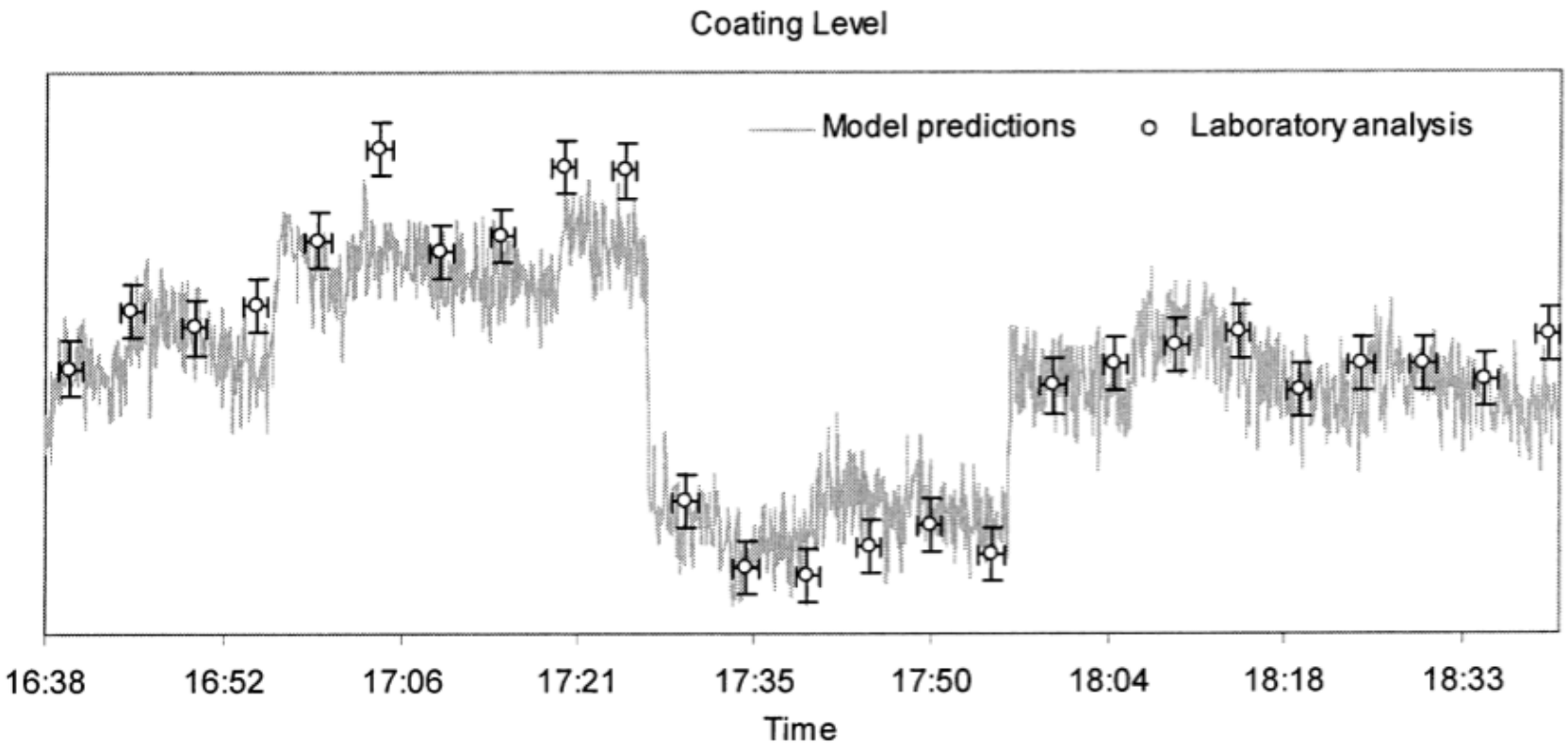


Figure 8. Schematic of the processes and imaging systems.

See http://dx.doi.org/10.1021/ie020941f

Example: snack food seasoning prediction

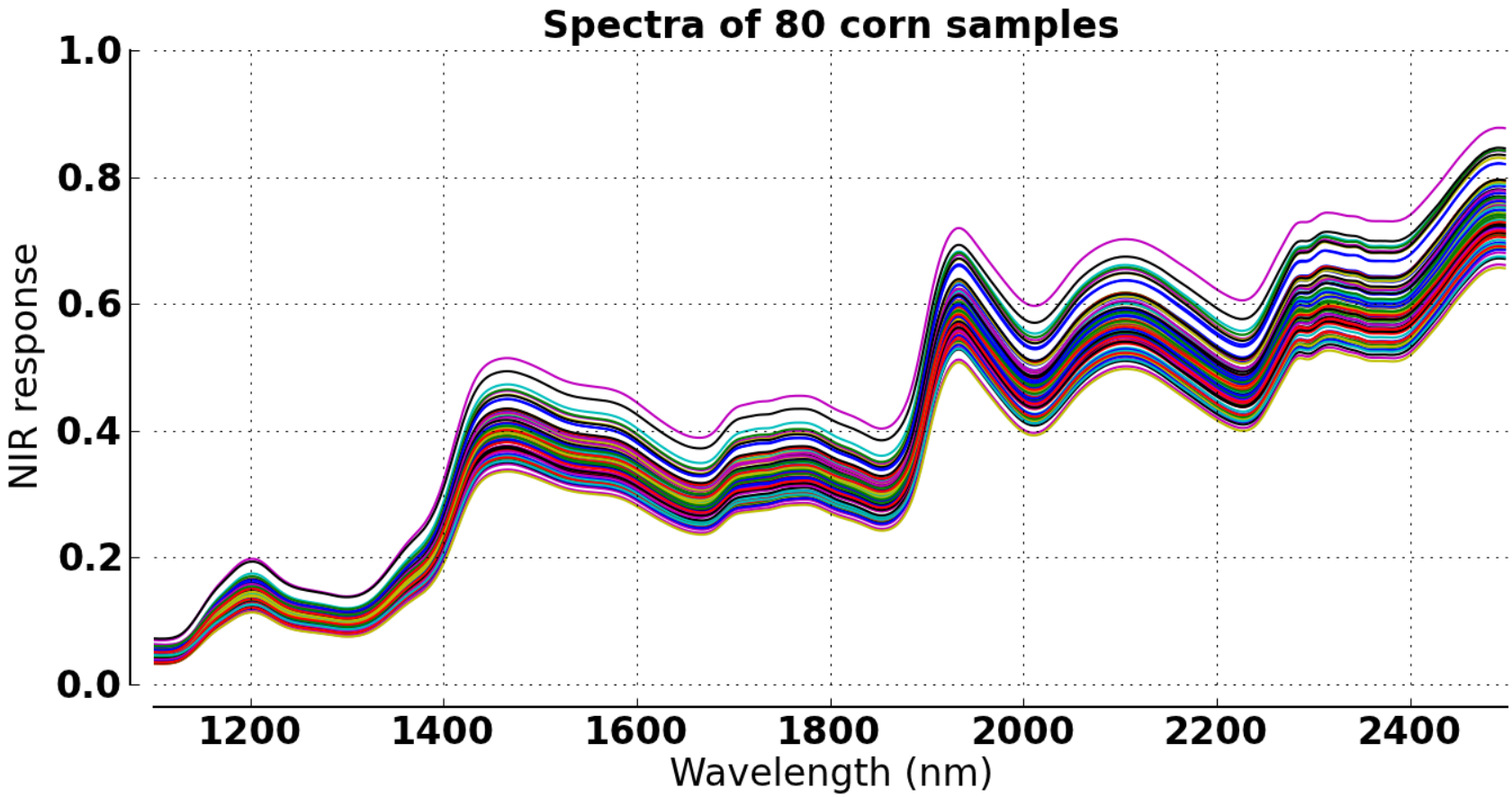


See http://dx.doi.org/10.1021/ie020941f

We will go into the details about this application in the section on "Image Processing"

Prediction of multiple properties of corn from NIR

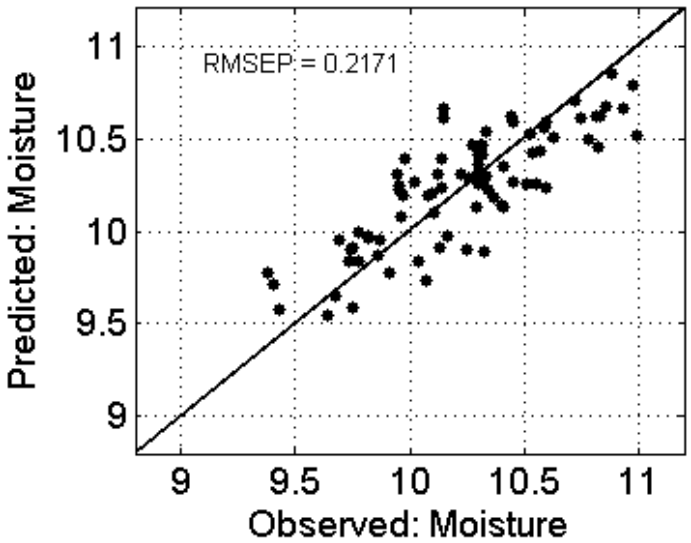
Source data are NIR spectra $\mathbf{X}$. There are 4 properties: y

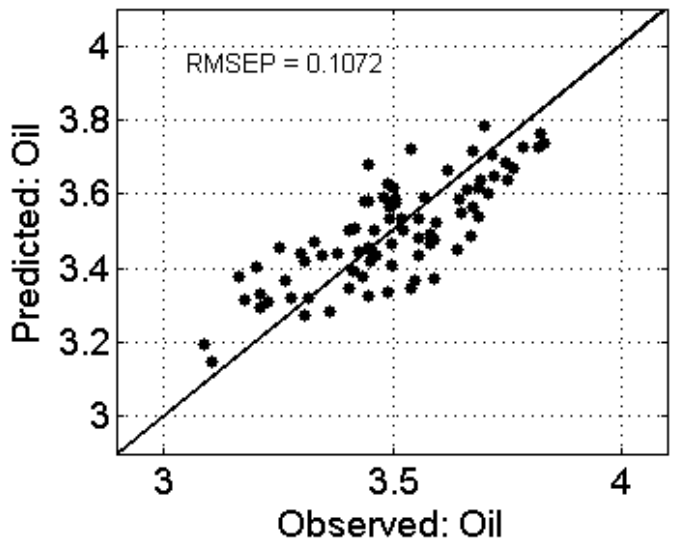


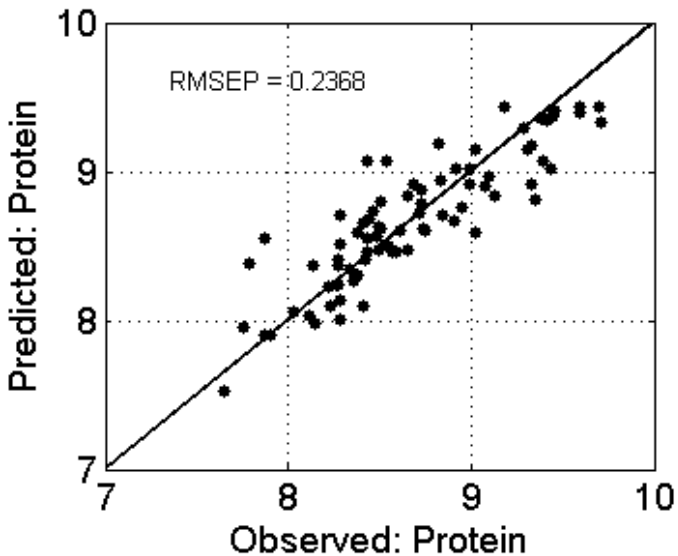
Source: http://eigenvector.com/data/Corn/index.html

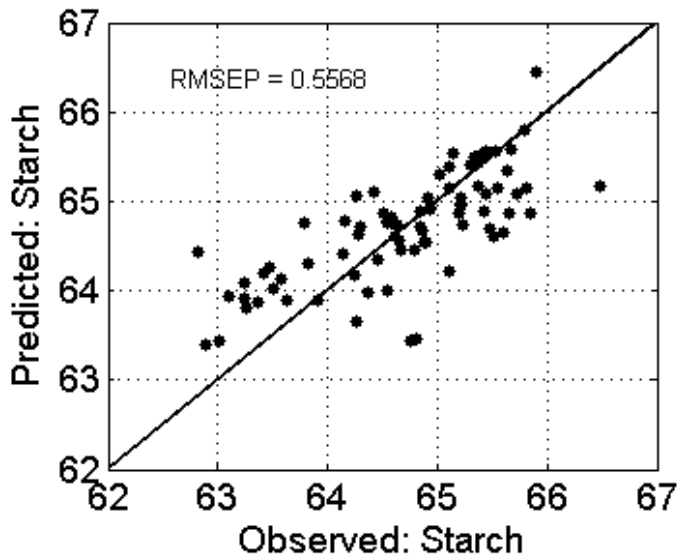
Prediction of multiple properties of corn from NIR

Prediction ability with 5 components:



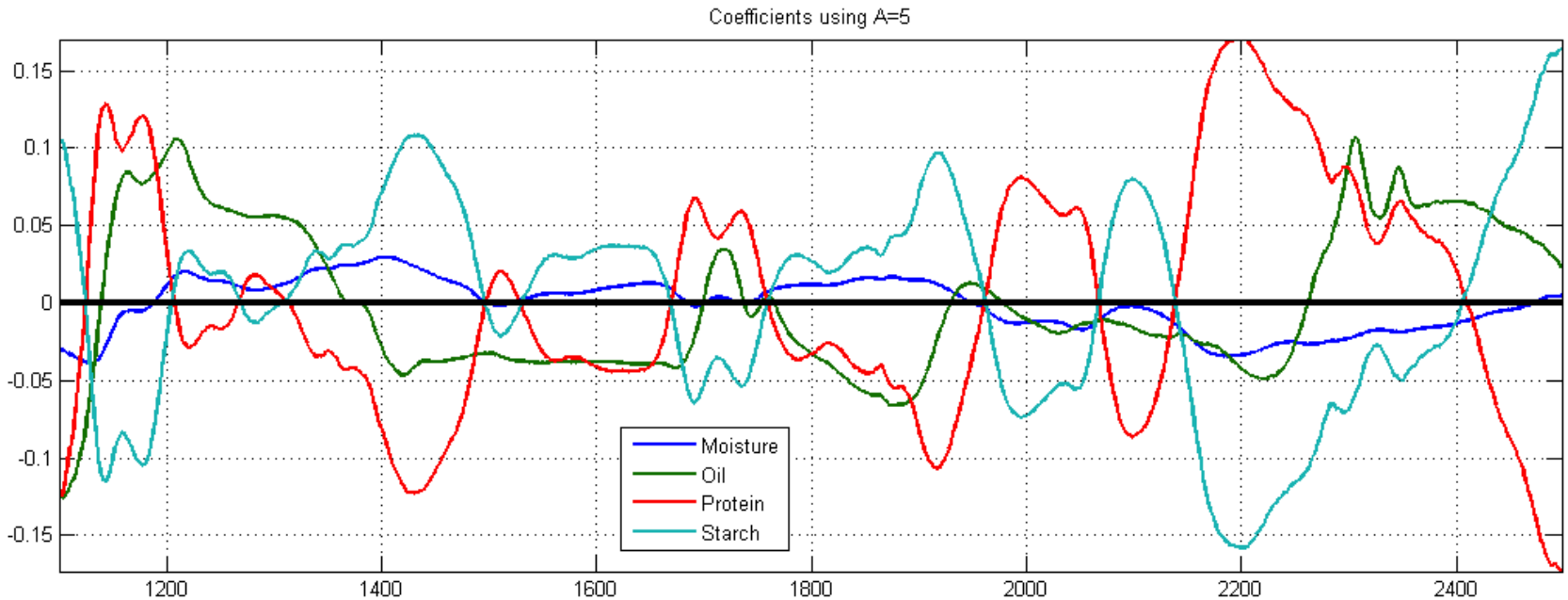






Prediction of multiple properties of corn from NIR

Coefficients, $\mathbf{b}$, for prediction: $y = \mathbf{x}_{\text{new}}\mathbf{b}$



Further advantages of soft sensors

Advantages include:

- That **X** data is more frequently available than **y** (real-time)
- **X** often has greater accuracy than the lab and sampling error in **y**
- Get a free process monitoring check from SPE and T^2 if you use PLS (not for other prediction models)
- Can handle missing values in **X** with PLS
- A model such as PCR or PLS will handle collinearity in **X**
- We also would like a realistic confidence interval for our prediction

Inferential predictions are only a *supplement* to the more expensive/time-consuming way of getting to the value of interest.

- We can reduce laboratory costs
- But, we should never eliminate the lab values though!

Calibration vs soft-sensors

Sometime you might see the term "multivariate calibration"

- Calibration: implies model is built from samples which are specifically acquired for the purpose of building the predictive model
- Soft sensors: built from historical, happenstance data (i.e. regular day-to-day data)

Calibration data is collected with the intention of building a predictive model.

Good reference: Martens and Næs

Concepts

Like we saw in process monitoring, there are 2 phases to building a soft sensor

1. Model building phase (training)
2. Testing the model's performance

Then you can go ahead and apply it on-line.

Concepts: judging model's performance

- **RMSEE** = root mean square error of estimation = $\sqrt{\frac{1}{N} \cdot \sum_i^N (y_i - \hat{y}_i)^2}$ $\longleftarrow$ when building the model
- **RMSEP** = root mean square error of prediction = $\sqrt{\frac{1}{N} \cdot \sum_i^N (y_{i,\text{new}} - \hat{y}_{i,\text{new}})^2}$ $\longleftarrow$ testing model on new data
- Bias = average offset = $\frac{1}{N} \cdot \left(\sum_i^N y_{i,\text{new}} - \hat{y}_{i,\text{new}} \right)$

These 3 metrics are easy to use: they are all in the original units of y, and we want them as small as possible.

Concepts: judging model's performance

Ideally: we have a completely separate test set for which we *minimize* its prediction error.

- Build predictive model
- Use model on testing set and calculate residuals
- Are residuals small enough? Acceptable overall performance?

We can't always afford a testing set, so the next best option: **test set switch**

- Split data in half
- Build model on part 1, test on part 2 to calculate RMSEP_2
- Build model on part 2, test on part 1 to calculate RMSEP_1

Once finished:

- Calculate average RMSEP: aim to minimize this.
- Finally use model built from all data.

Exercise

Time for a calibration exercise ("sawdust")

Cross-validation for soft-sensors

Why is cross-validation **not suitable** for *soft-sensor* models?

- Can it be made more suitable?

Helpful references on soft sensors

1. Systematic and robust procedure described by: Lin *et al.* (paper 107)
2. Kresta, Marlin and MacGregor: Development of inferential process models using PLS
   - Describes some cautions when using soft sensors

Systematic procedure to **build** a soft sensor (phase 1)

1. Start with available data that is representative of process operation
2. **Remove outliers***: plant shutdowns, gross outliers
3. Build initial model
4. Remove outliers in T^2 and SPE
5. Rebuild model (may need to iterate back to step 2)
6. From here on: aim is to maximize RMSEP on a completely separate testing set
7. What data transformations might improve predictions? See next section on "Advanced Preprocessing"
8. Drop out noisy variables (small weights, small VIP)
9. Iteratively build your models until you get desirable performance
10. Your final model should make physical sense
    - known important variables should have large weights

* Try to ensure you can use the same cleaning routine on-line (phase 2)

Online procedure for a software sensor I

Phase 2: using a soft sensor online

1. Collect the input variables required
2. Do univariate cleaning and checks (the same as when building the model)
3. Perform any calculations, feature extraction, etc, on variables that will create new columns
4. You should now have assembled your vector: $x_{\text{new, raw}}$ (might have missing values)
5. Preprocess this vector (the same as when building the model), to get $\mathbf{x}_{\text{new}}$
6. Project this vector on to the latent variable directions, $\mathbf{W}^*$, to get $\mathbf{t}_{\text{new}} = \mathbf{x}_{\text{new}}\mathbf{W}^*$
7. Calculate projected $\hat{\mathbf{x}}_{\text{new}} = \mathbf{t}_{\text{new}}P'$
8. Calculate residuals: $\mathbf{e}_{\text{new}} = \mathbf{x}_{\text{new}} - \hat{\mathbf{x}}_{\text{new}}$

Online procedure for a software sensor II

9. Calculate SPE from $\mathbf{e}_{\text{new}}$
   - Below the limit: *continue*
   - If not: investigate contributions* to diagnose problem; *you really shouldn't continue*
10. Calculate Hotelling's T^2 from $\mathbf{t}_{\text{new}}$
    - Below the limit: *continue*
    - If not: investigate contributions* to diagnose problem; *continue with caution* if not a huge outlier (subjective)
11. Make your predictions: $\hat{\mathbf{y}}_{\text{new}} = \mathbf{t}_{\text{new}} C'$
12. Un-preprocess your predictions back to real-world units
13. Check your prediction quality:
    - Run your prediction model in parallel to lab values initially
    - Check model predictions against lab values to determine if it needs updating.

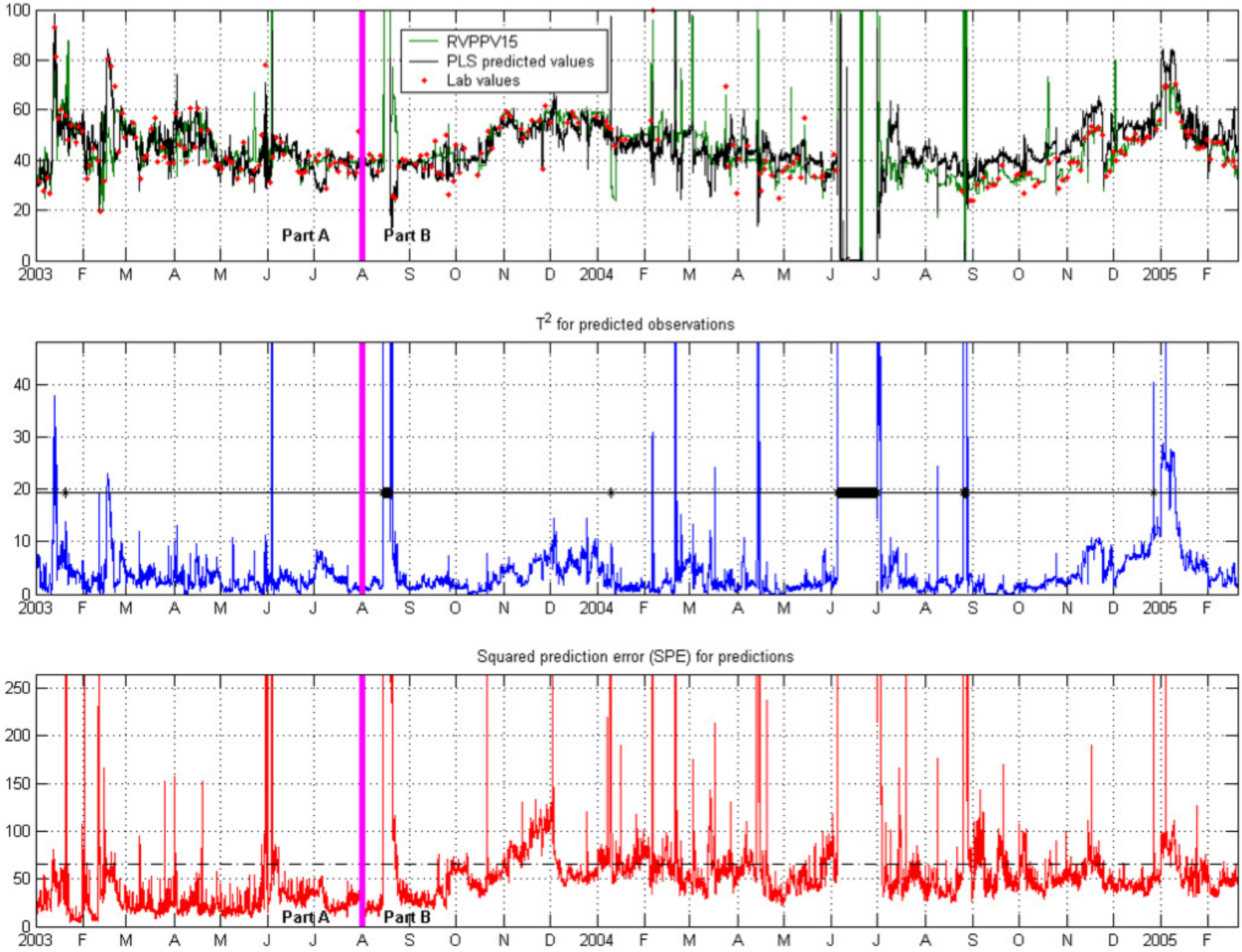
*** Note**: a useful procedure if only a handful of variables are flagged in the contributions: set them as missing values and repeat the prediction. If this doesn't work, then you will need to rebuild the soft sensor.

Dealing with poor online prediction performance

Predictions from the model are valid as long as the correlation structure is consistent.

- Track SPE: above the 95% limit indicates that things have changed (see next slide)
- Track T^2: increased throughput? Adjust mean-centering and perhaps the scaling
- Drift in some variables*: adjust mean-centering vector
- Still not solved: may need to rebuild model, especially after major events (plant shutdown and clean; new equipment installed)

This is another reason why we
* We will deal with this topic in a later class on adaptive kernel methods (this is a good project topic)



Degraded performance later on (part B). Project for someone?

Dealing with multiple product grades

Many processes produce multiple products on the same line

- Slightly different properties
- Require predictions for the same (quality) variable of interest
- Can we have one predictive model for multiple product grades? Another potential course project

Key objective:

- Switch between grades with minimum time

Caution: Missing values

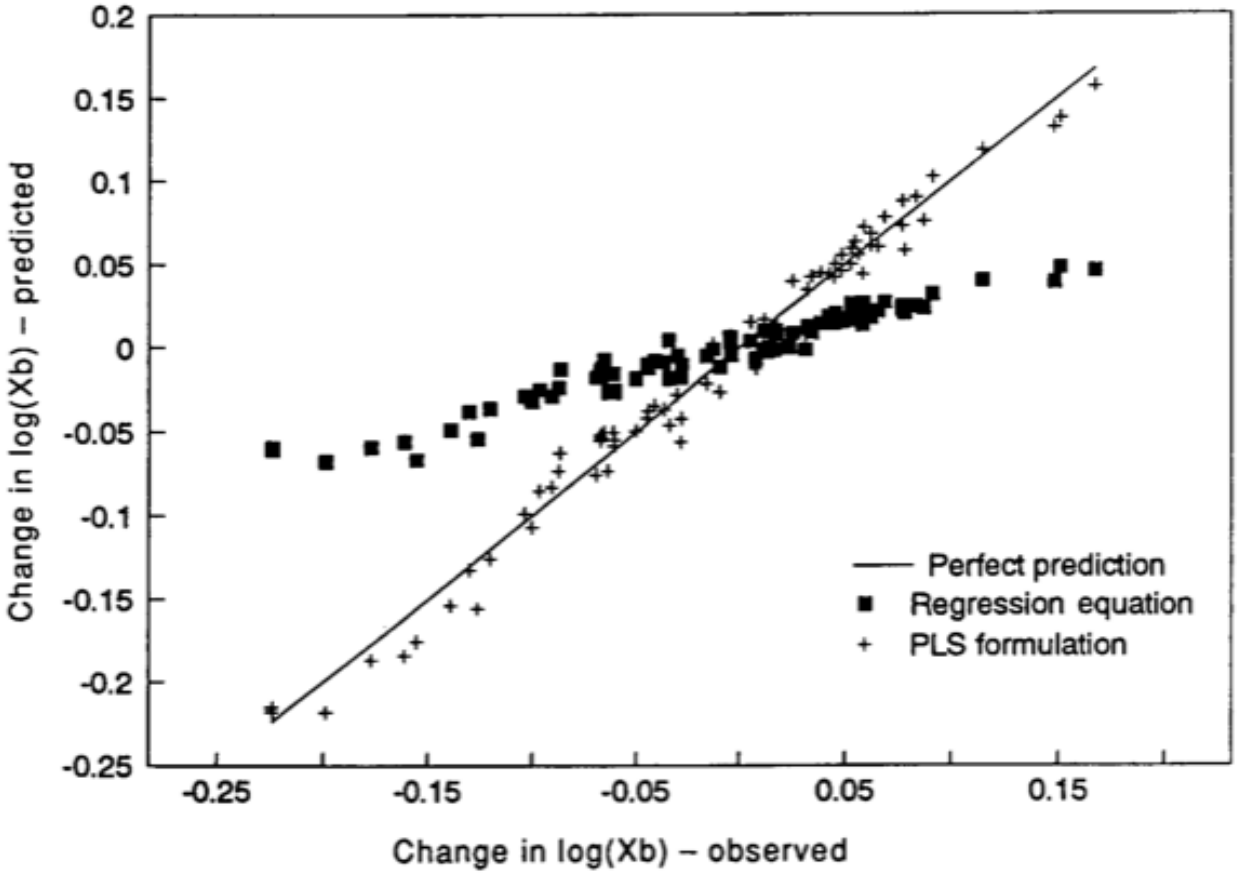
- Sensors go offline from time-to-time
- Often missing data are systematic: uneven sampling frequency between variables

The last case should be carefully checked:

- Try different missing value algorithms (project topic!)
- Build model only on rows where most data is available and compare to full model.

Caution: Missing values

- From paper by Kresta, Marlin and MacGregor



- Regression model replaced missing value by the mean
- PLS model used usual missing data handling

Caution: Have adequate variation in training data

- Have adequate variation in the data

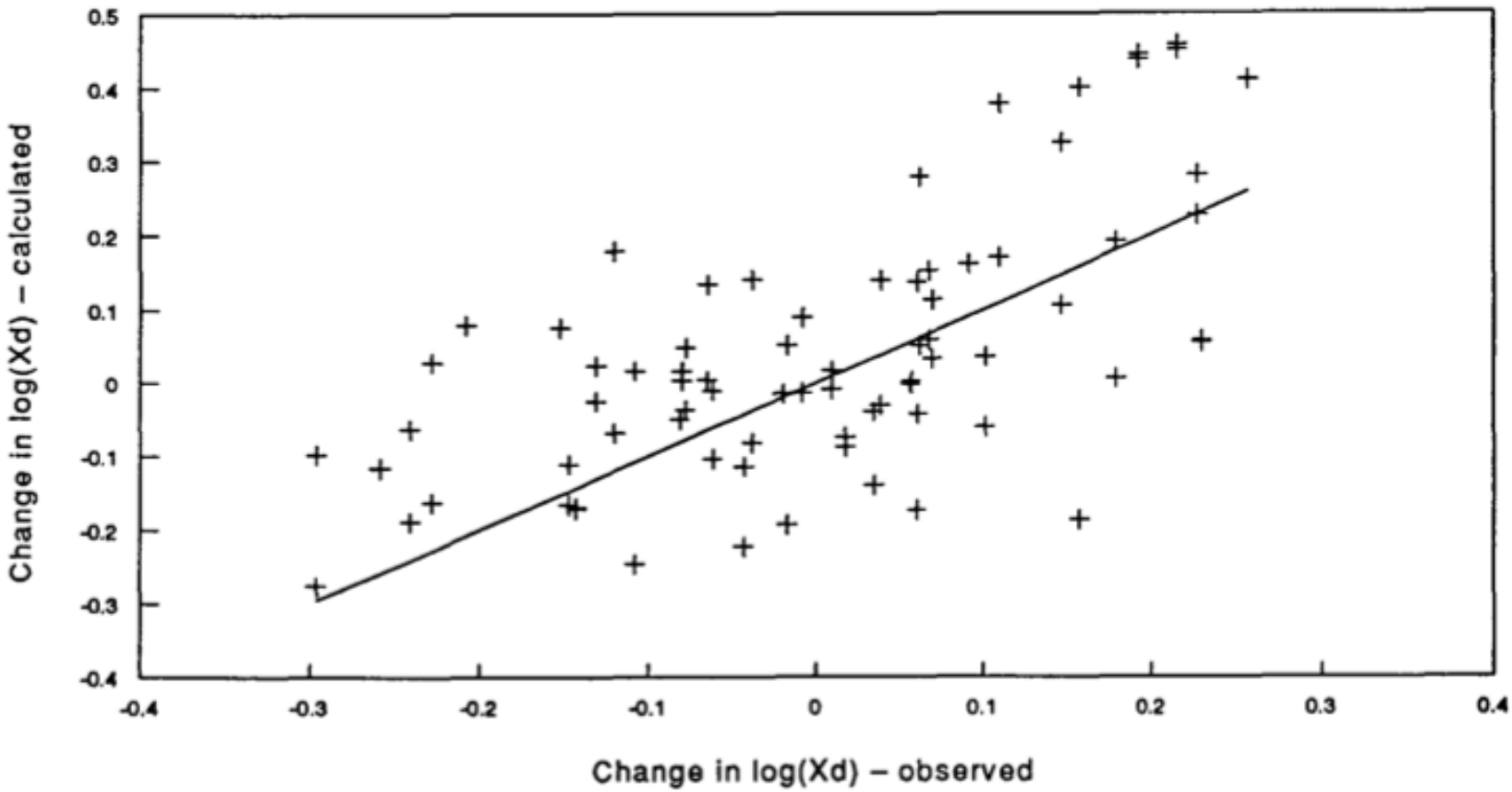


Fig. 6. Prediction of the methanol composition in the distillate for the MAW column using a model developed from a data set containing only variation in the manipulated variables. This shows the degradation of the prediction when an improper data set is used.

Caution: Have adequate variation in training data

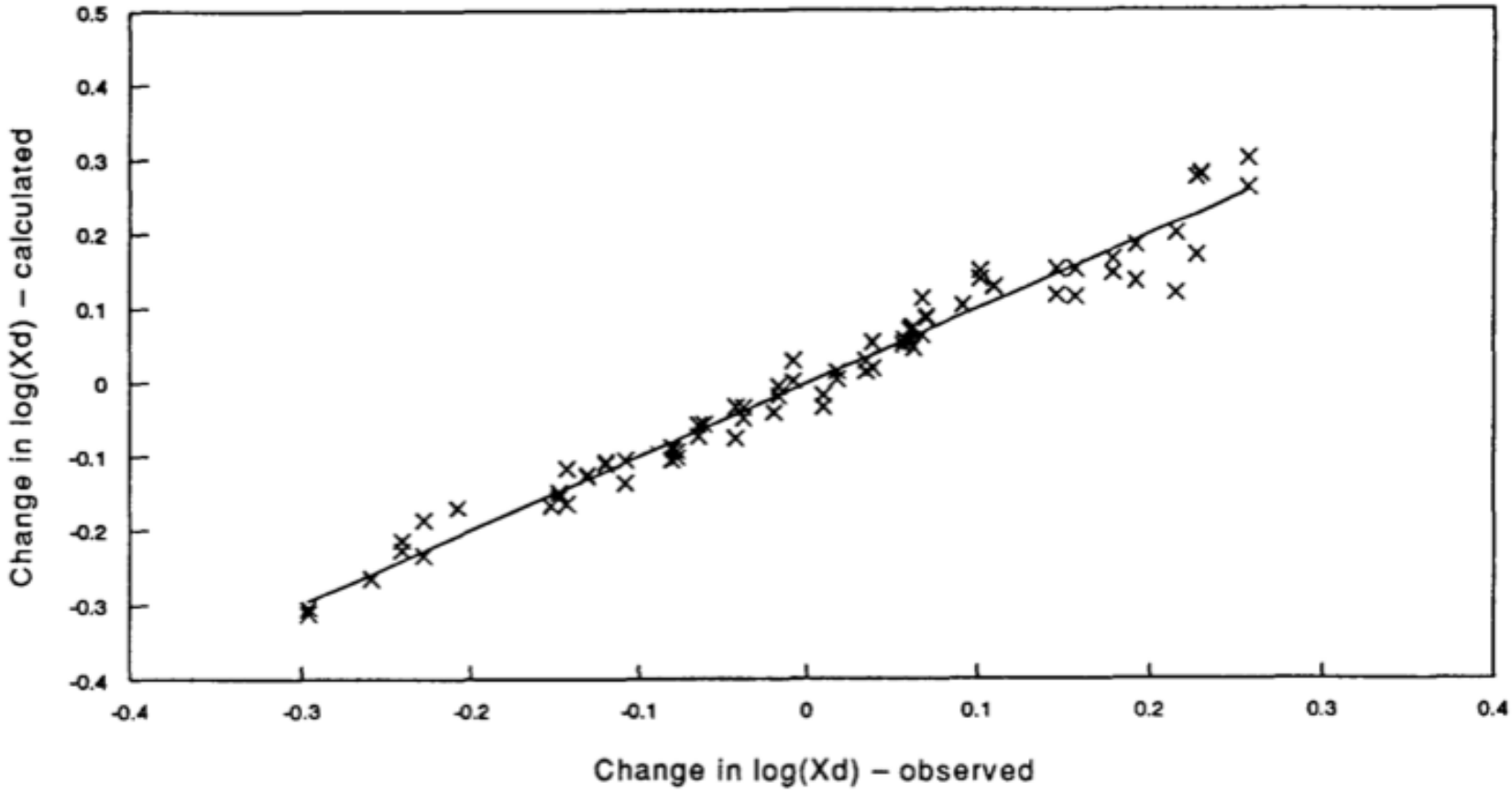
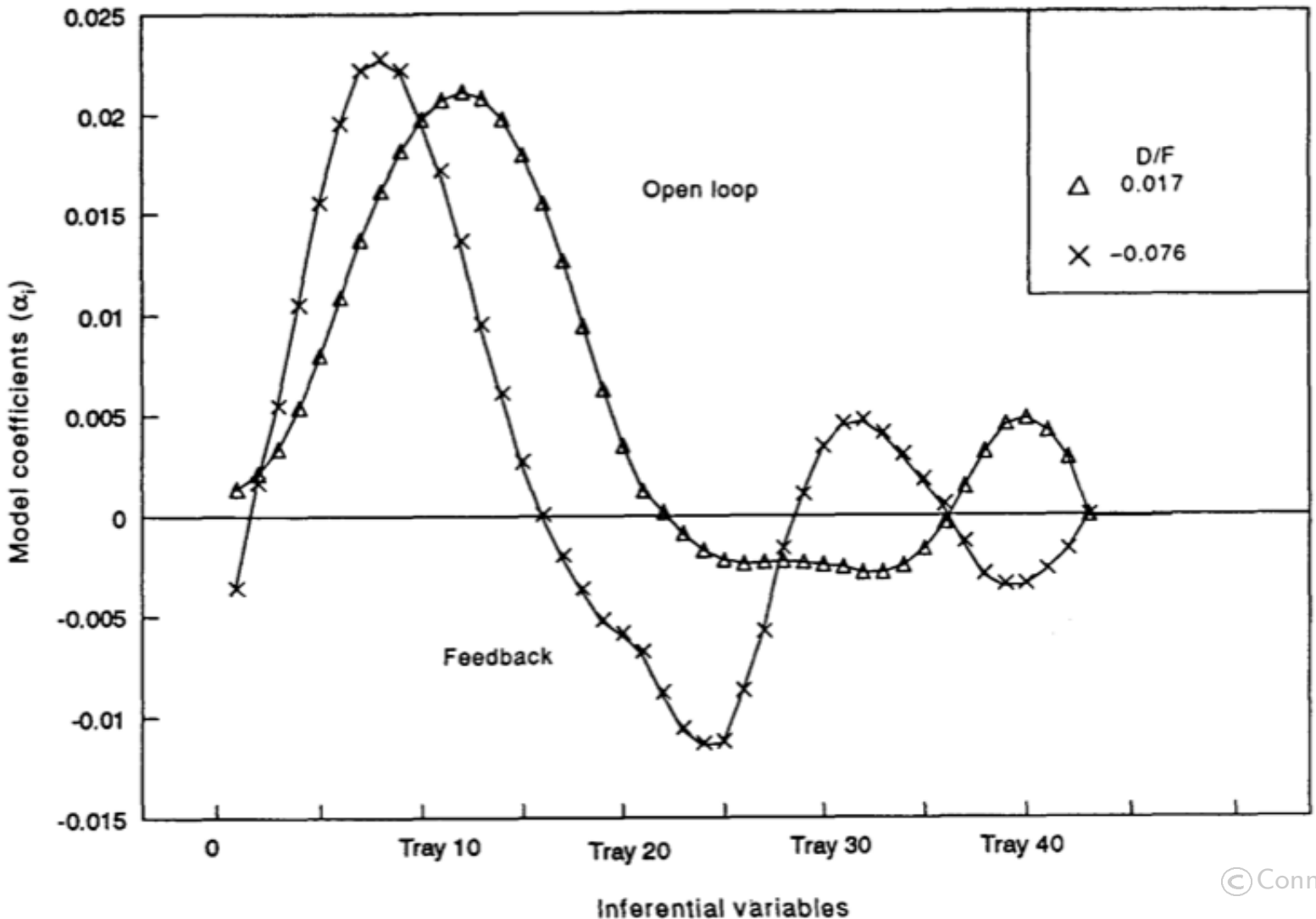


Fig. 5. Prediction of the methanol composition in the distillate for the MAW column using a model developed from a proper data set, one containing typical variation of both manipulated and disturbance variables.

- The model-building data must represent all expected conditions under which it will be applied.

Caution: Dealing with feedback control

- Feedback control changes the correlation structure among variables
- Recall: if correlation structure changes - **rebuild model**



A model per y-variable or one PLS model

Rule of thumb:

- Do a PCA on the Y-variables
- If there is strong correlation between some of them, build a single PLS for the correlated Y's
- Build a separate model for Y's that are not correlated with each other

Advanced Preprocessing

In this next section we will look at data preprocessing

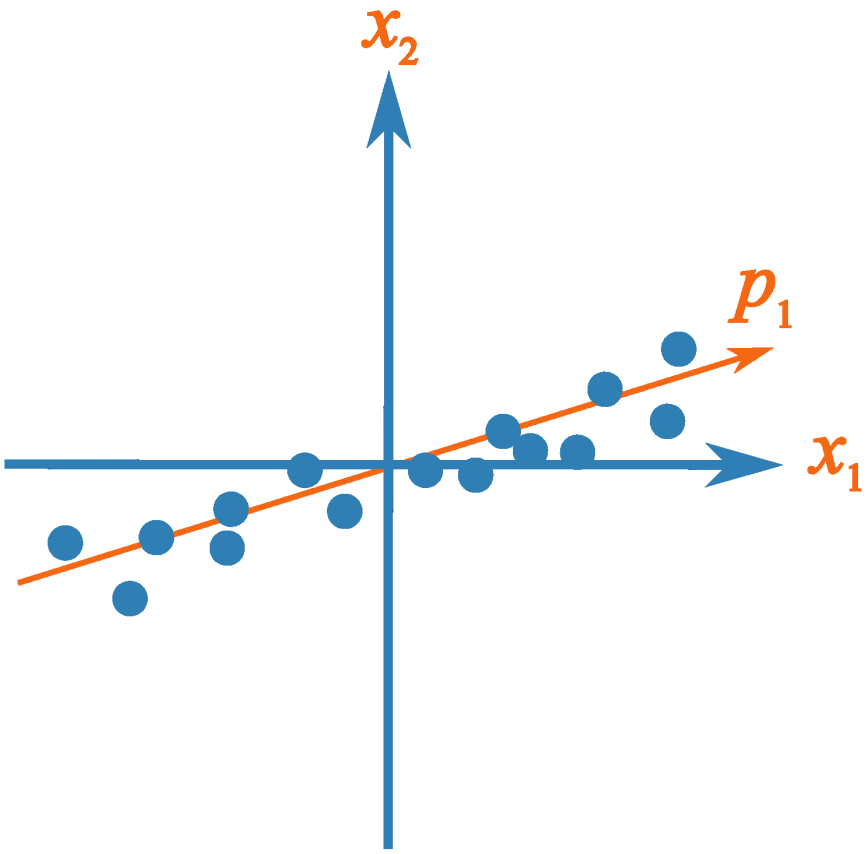
- Centering
- Scaling
- Block scaling
- Trimming and Winsorizing
- Univariate transformations
- Calculated variables
- Integer and categorical variables (qualitative variables)
- Process dynamics handled by lagging columns in **X** and/or **Y**
- Dealing with nuisance variation
- Meaningful feature extraction

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

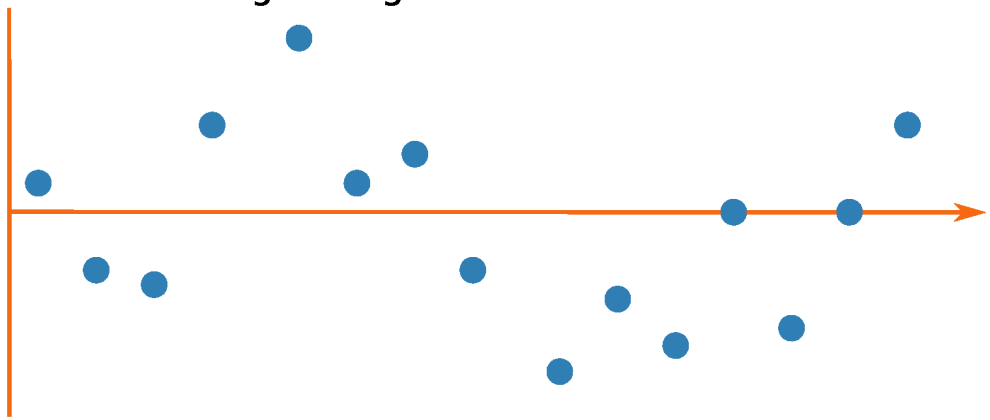


- If no prior knowledge: scale each column to unit variance
- Need to emphasize/deemphasize a variable?
  - First scale all columns to unit variance
  - Then upscale/downscale the column(s) as appropriate

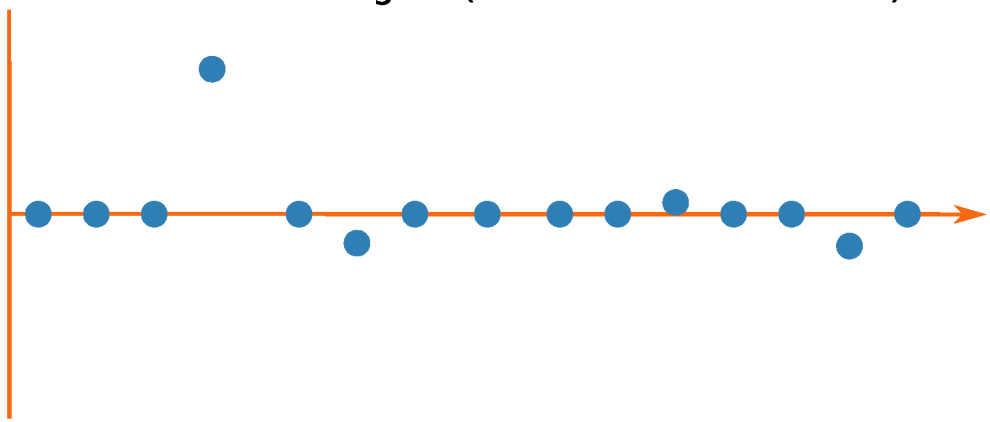
Scaling

Be careful of inflating noise in variables with little signal

Variable with good signal



Variable with little signal (low standard deviation)

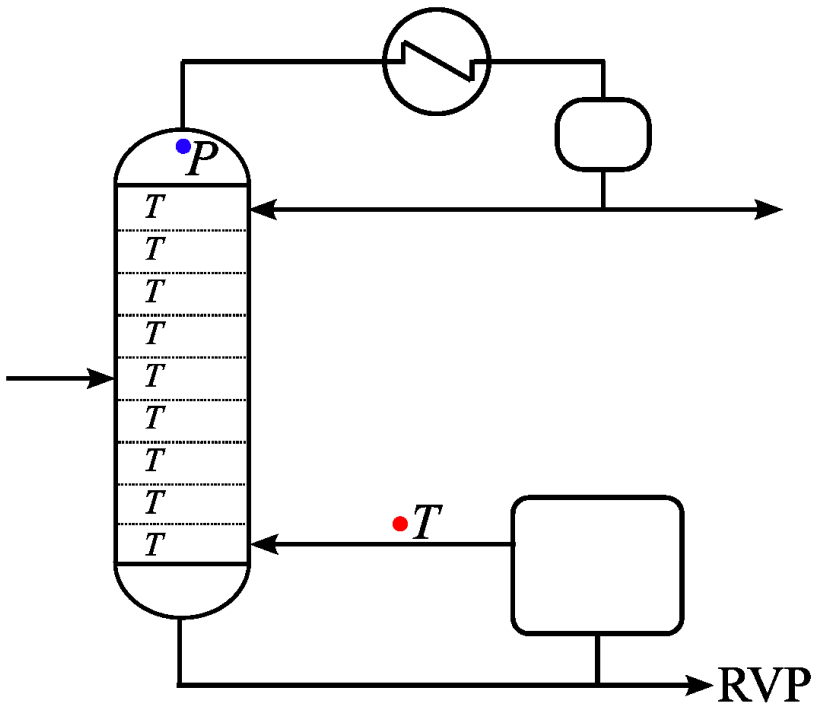


- Use robust scaling instead (scale by MAD)
- Heuristic: don't scale x_k if $\text{stdev}(x_k) < 4 \times$ its measurement noise

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

Block-scaling, or group-scaling

Distillation column example: 80 tags



- 50 tray temperatures
- 30 other tags available

What will be the prominent direction in PCA or PLS model?

Block scaling:

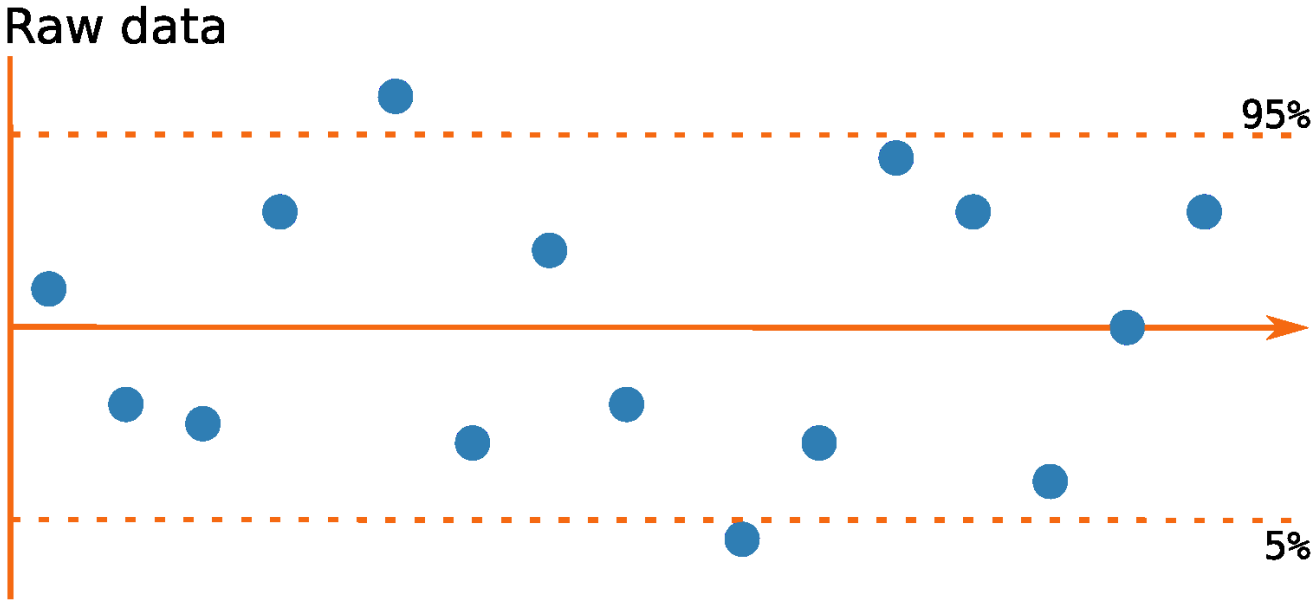
- Do normal preprocessing
- Then divide group of tags (called a block of variables) by $\sqrt{K_b}$

Robust centering and scaling

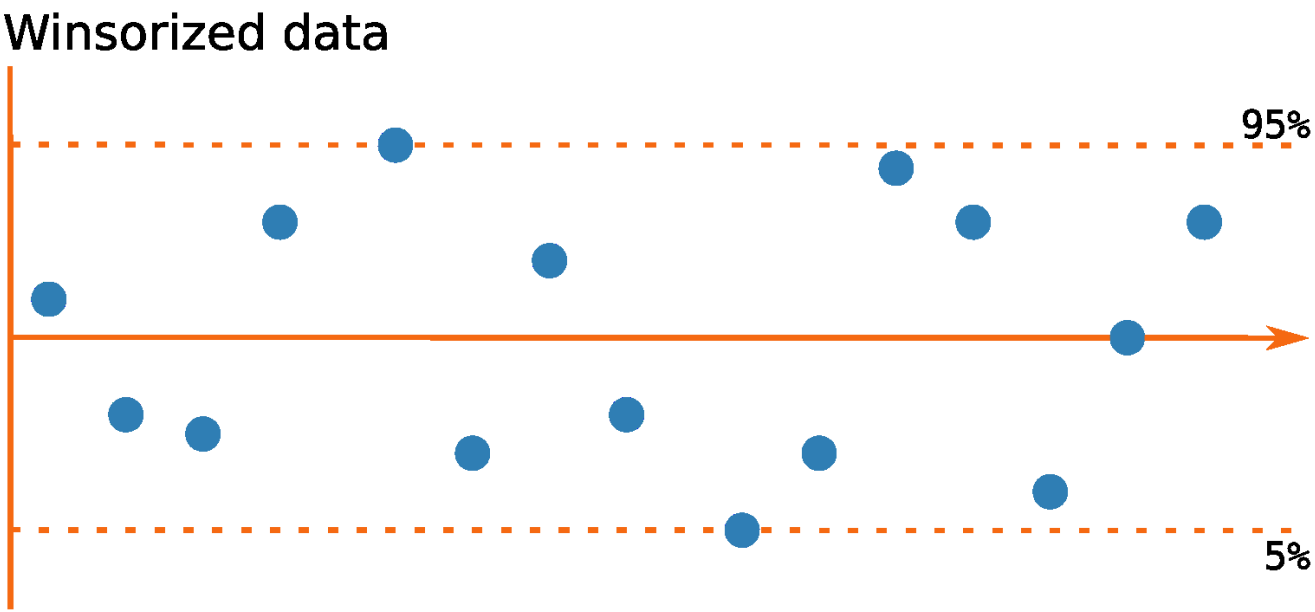
- Use robust preprocessing to emphasize outliers
- Interesting project topic
  - Does robust preprocessing complicate or help the subsequent PCA/PLS model-building?
  - What about robust preprocessing followed by robust PCA or robust PLS?
  - What if we modify the NIPALS iterations with a robust alternative?
    - Use the hat-values to down-weight outliers?
    - Several robust LS methods: e.g. Tukey, quantiles-and-medians, and others
    - Will the NIPALS algorithm still converge?

Very readable paper: Filzmoser and Todorov, 2011: "Review of robust multivariate statistical methods in high dimension"

Dealing with outliers: winsorizing



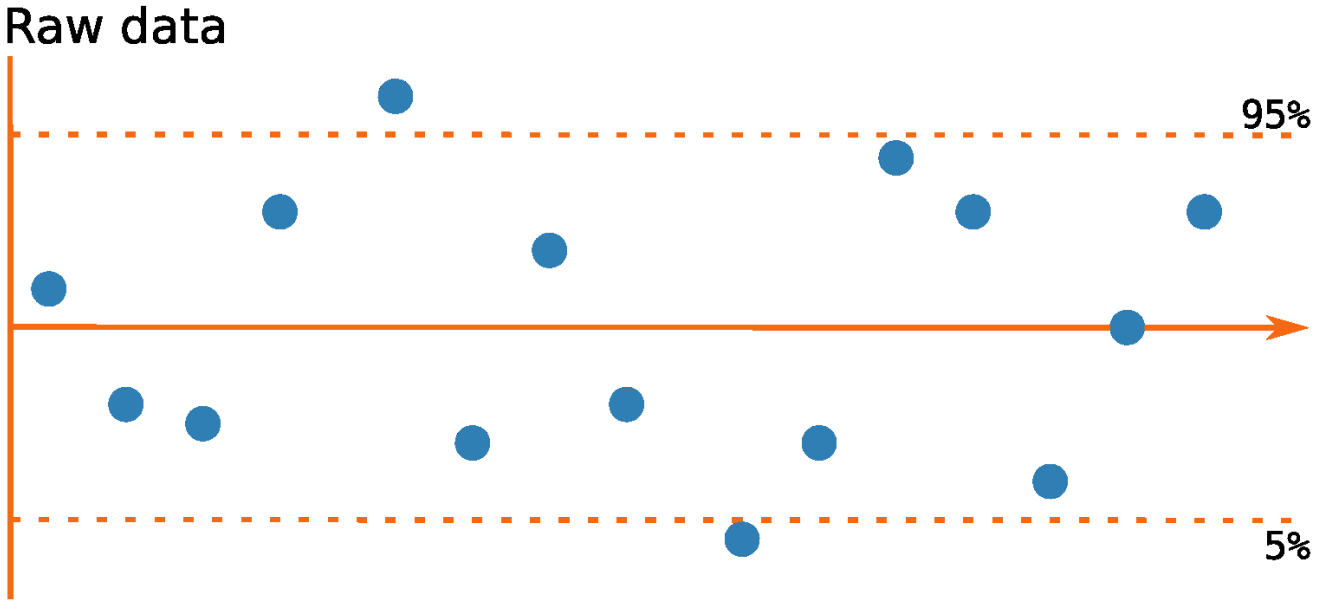
Replace outliers beyond a chosen α level at their respective bounds



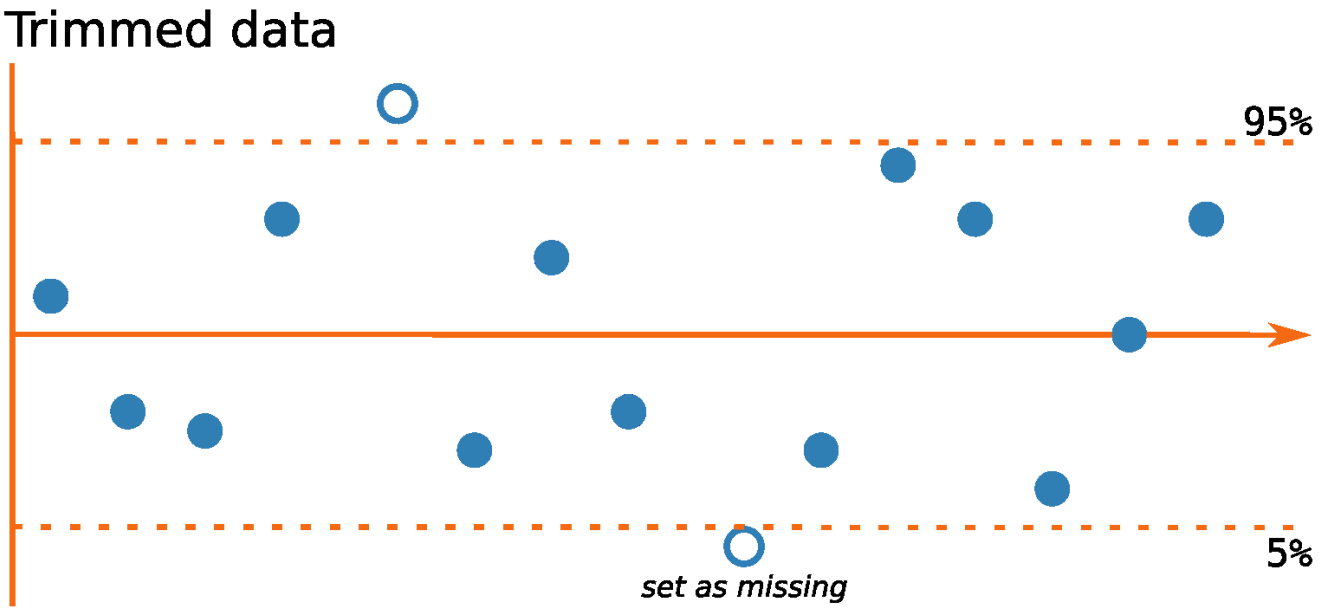
e.g. $\alpha = 5\%$ shown here

Can break multivariate relationships between variables

Dealing with outliers: trimming



Convert outliers beyond a chosen α level to missing values



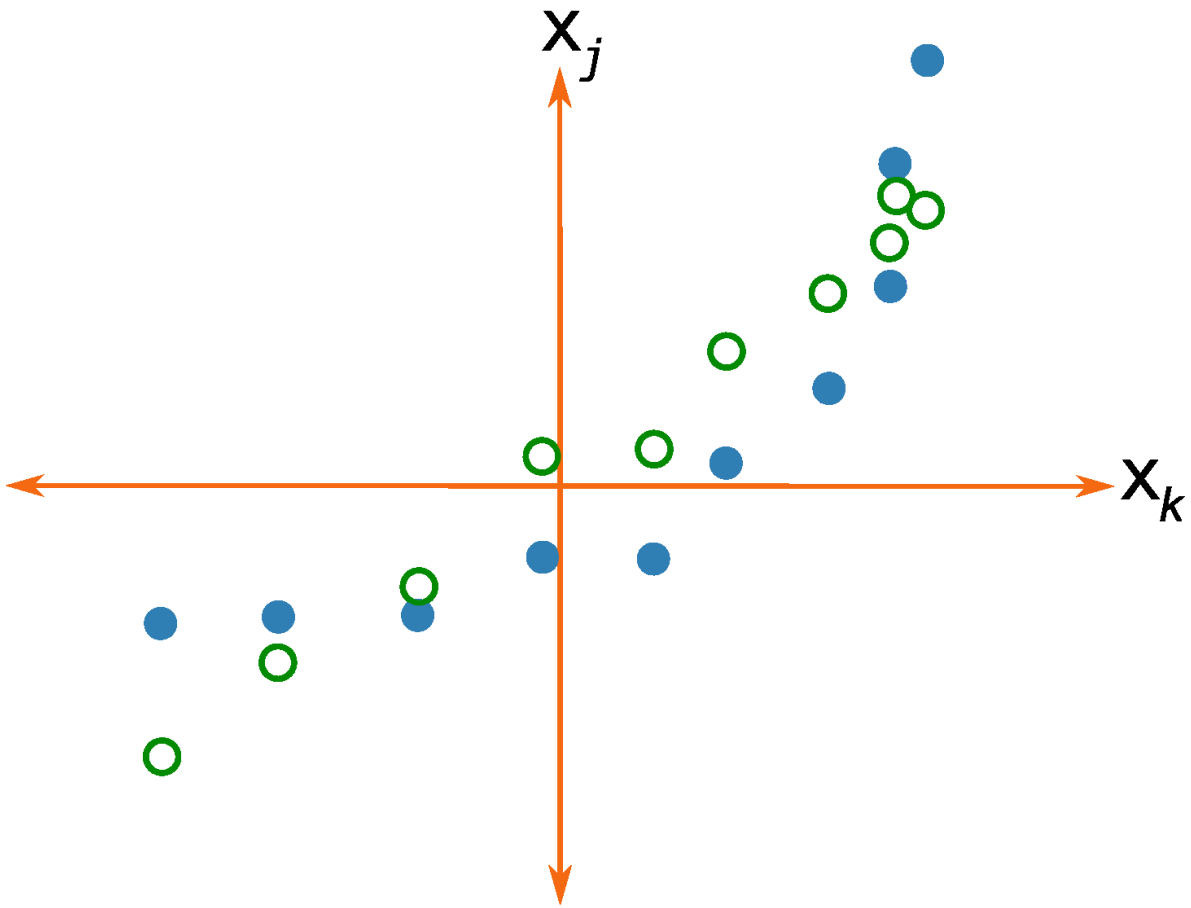
e.g. $\alpha = 5\%$ shown here

Better (more honest) alternative to winsorizing

Univariate transformations

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

- Original data, then centered and scaled
- Transformed data, then centered and scaled



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:

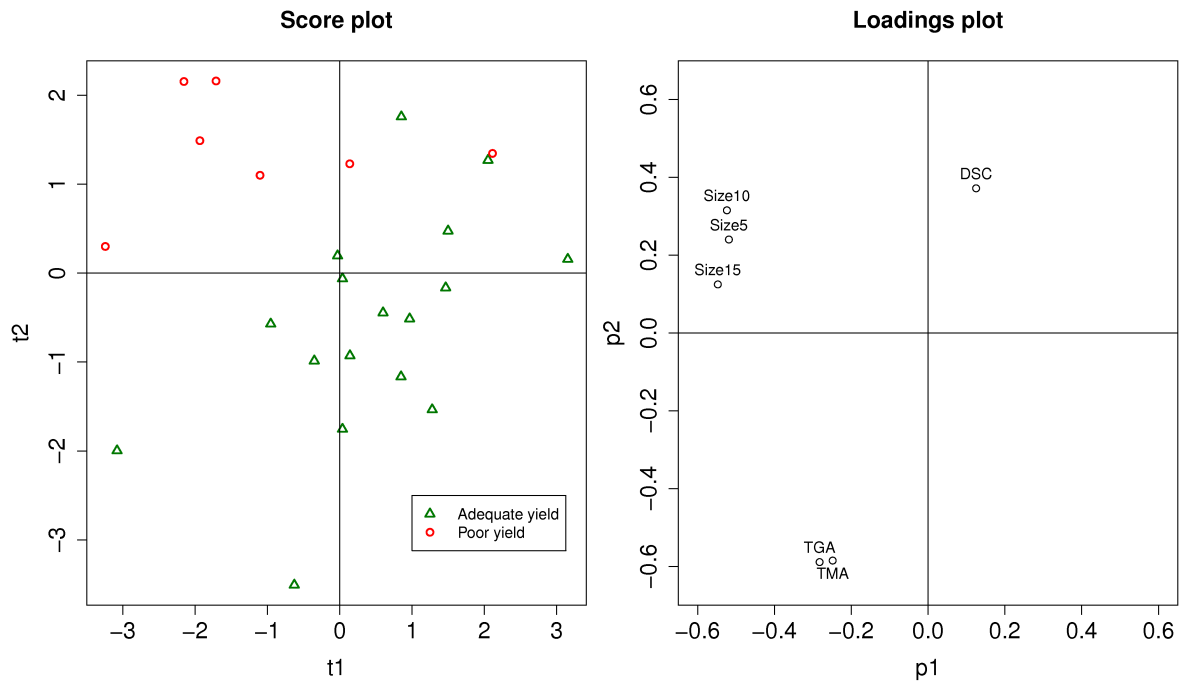
- Add columns from first-principles equations, e.g. Antoine equation: $\log(\text{RVP}) = \log P - B\left[\dfrac{1}{C} - \dfrac{1}{T+C}\right]$

Handling qualitative variables

Binary variables

- (yes/no) **or** (on/off) **or** (present/absent)
- Also called "dichotomous variables"
- Included and used like any other variable
- Centering and scaling affected by the relative number of rows from each category
- *illustration on the board*

Or just use variable to colour-code scores by:

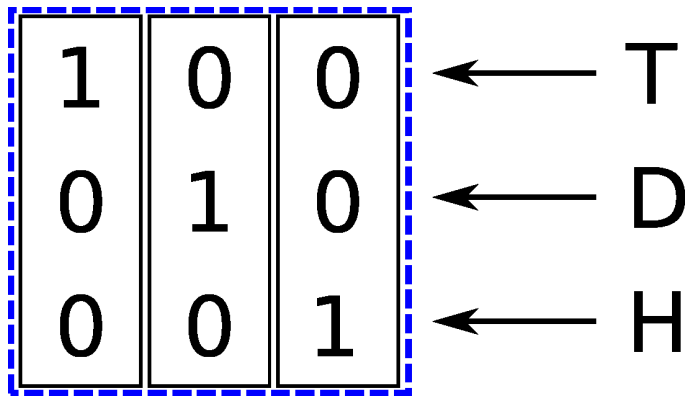


Handling qualitative variables

Unordered indicators must be expanded into extra columns

- aka "dummy variables", or "categorical variables"
- can be done with **X**-space and/or **Y**-space variables

e.g. reactor T, reactor D, reactor H



Should then block scale the group of columns, especially if number of levels is high

We will use this in the class on "Classification"

Ordered categorical variable: ordinal variable

e.g. Primary school, High school, College, Bachelor's, Masters

- Can convert them to a single column of integers. e.g.
  - 1 = Primary school
  - 3 = College
  - 5 = Masters
- You may choose to leave them as a single column then
  - e.g. months of the year: Jan=01, Feb=02, *etc*
- Loadings interpretation: same as a continuous variable
- As a predictor in the **Y**-space: round prediction to closest integer

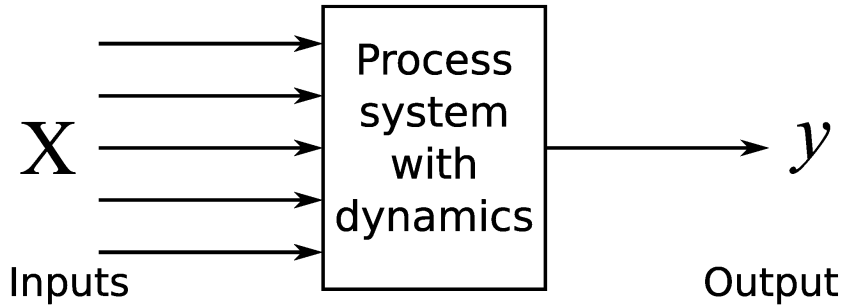
Caution: In many cases the gap from say 1 to 2 is not the same as the gap from say 3 to 4.

Rather expand into columns then.

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: $\frac{y(s)}{x(s)} = \frac{K}{\tau s + 1}$
- Time domain: $\tau \frac{dy}{dt} + y(t) = Kx(t)$
- Take samples Δt time units apart:

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

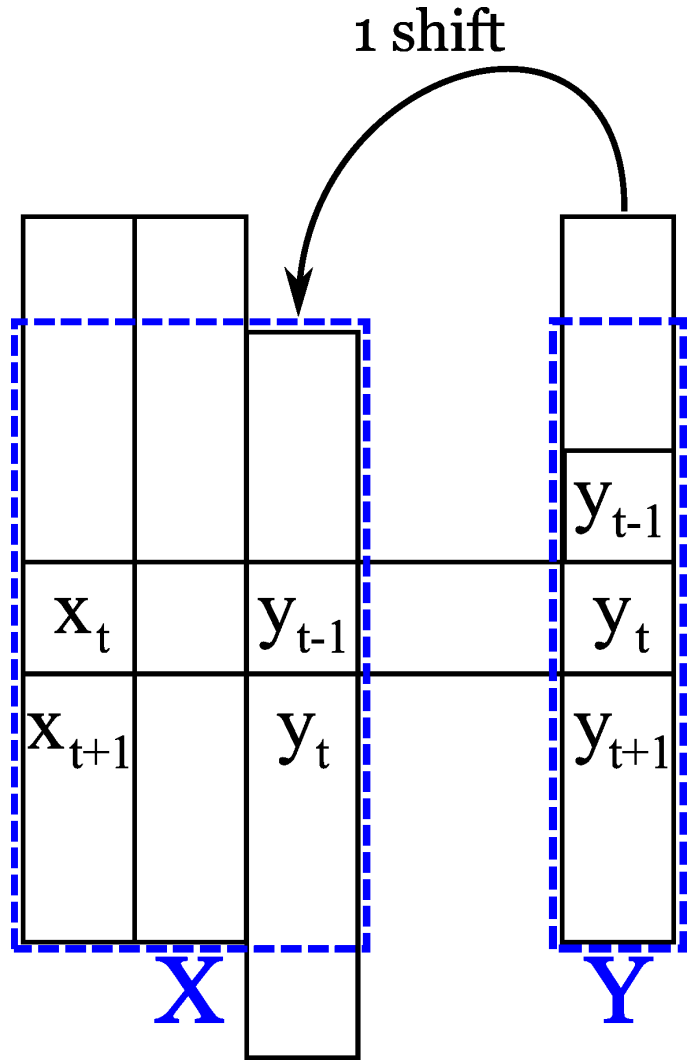
General form:

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

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

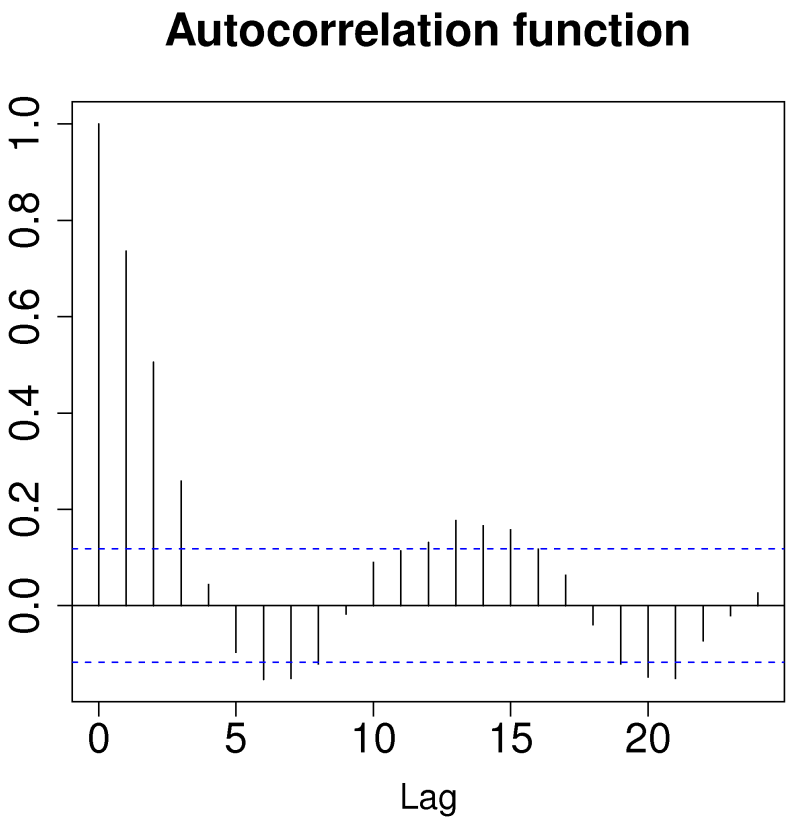
Time series example: lagging Y's

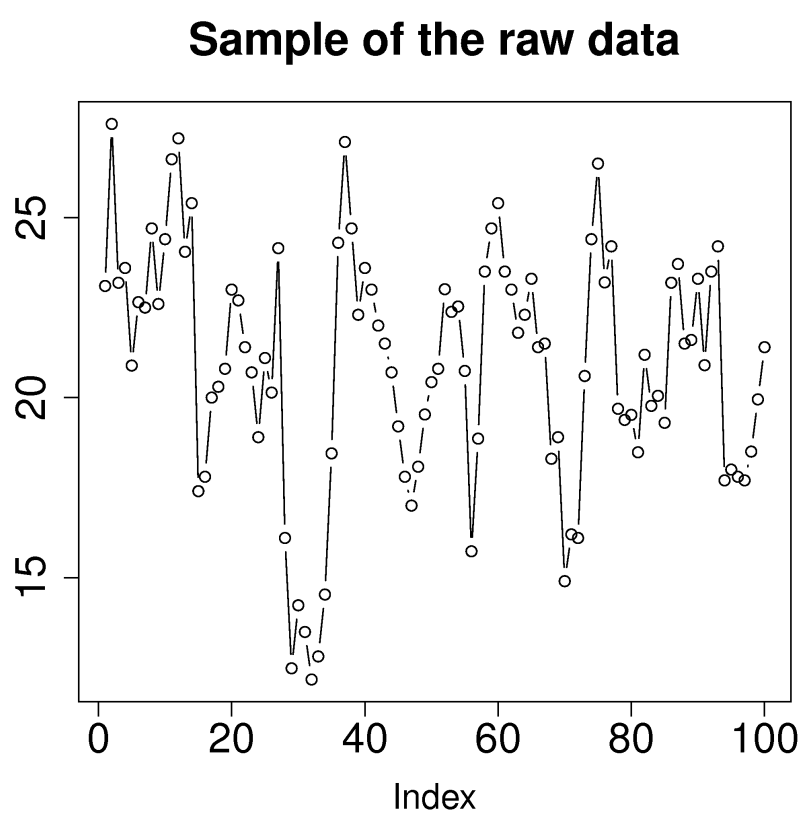
- How to approximate this with a latent variable model?
- Copy and shift columns to get consistency within a row
- So add columns to **X** to get the time-history of y
- $y(t) = a_1 y(t-1) + b_0 x_t + \ldots$



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)`

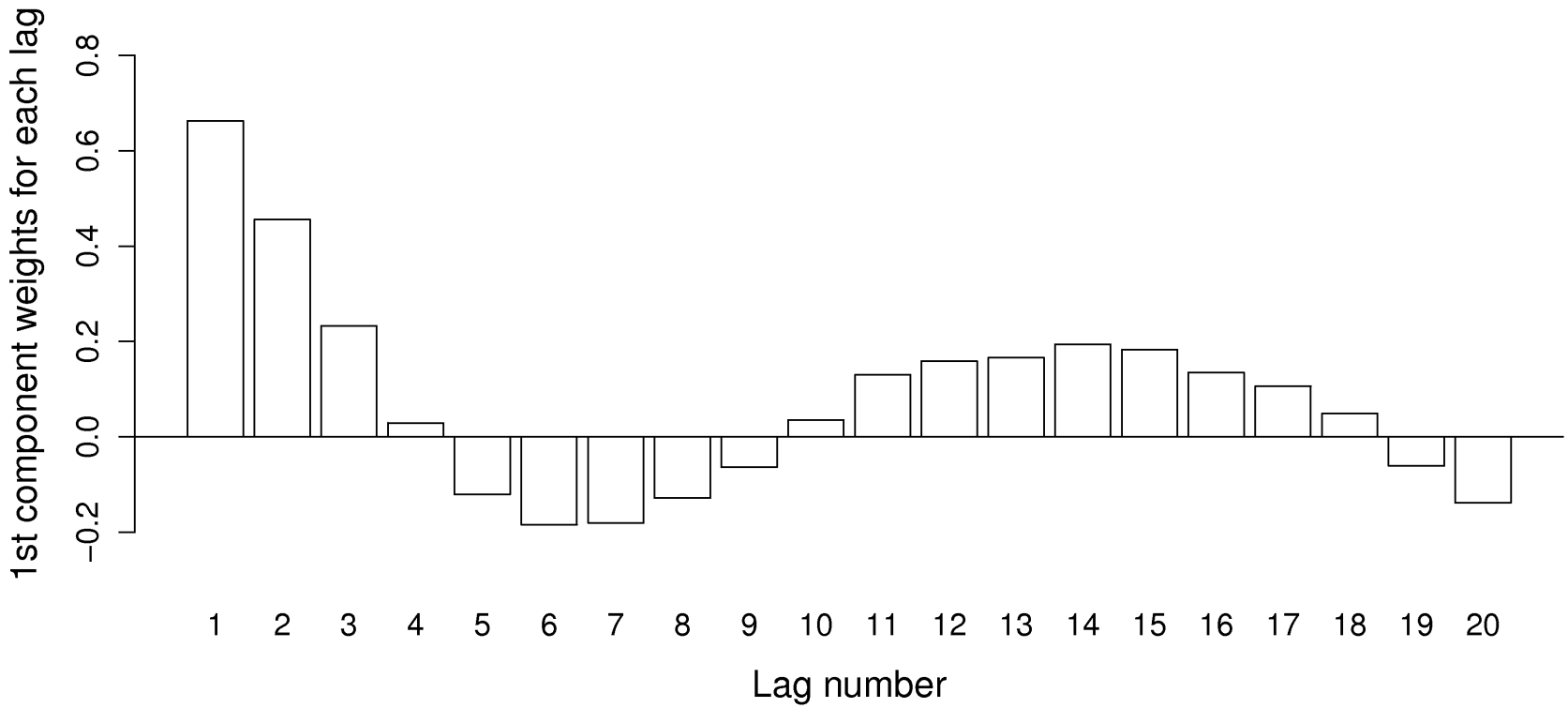




Time series example: lagging Y's

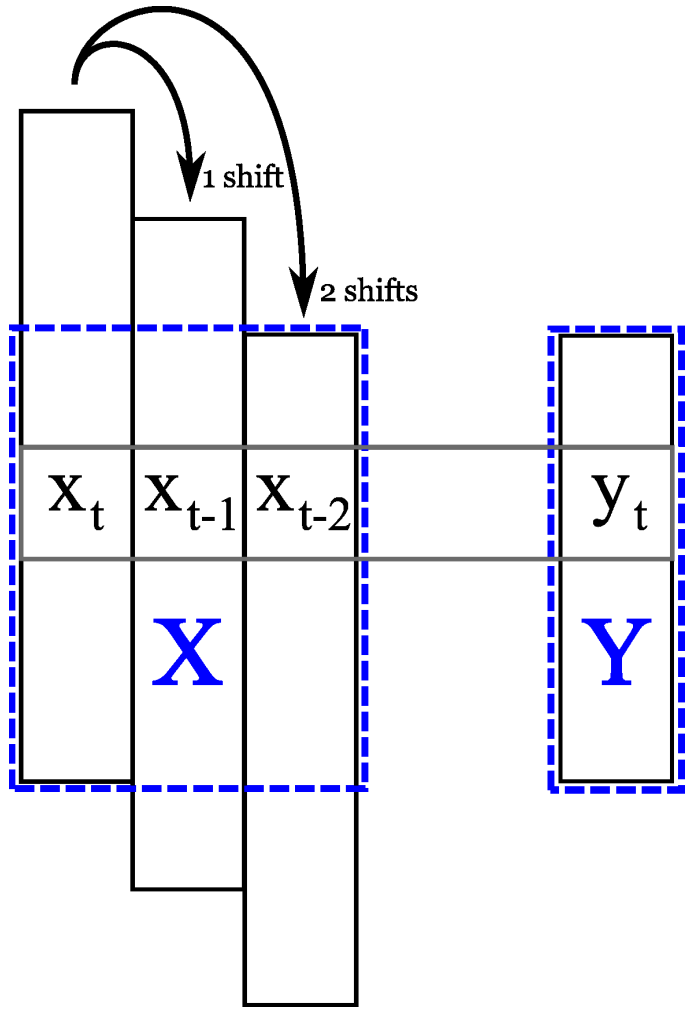
Other approaches:

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

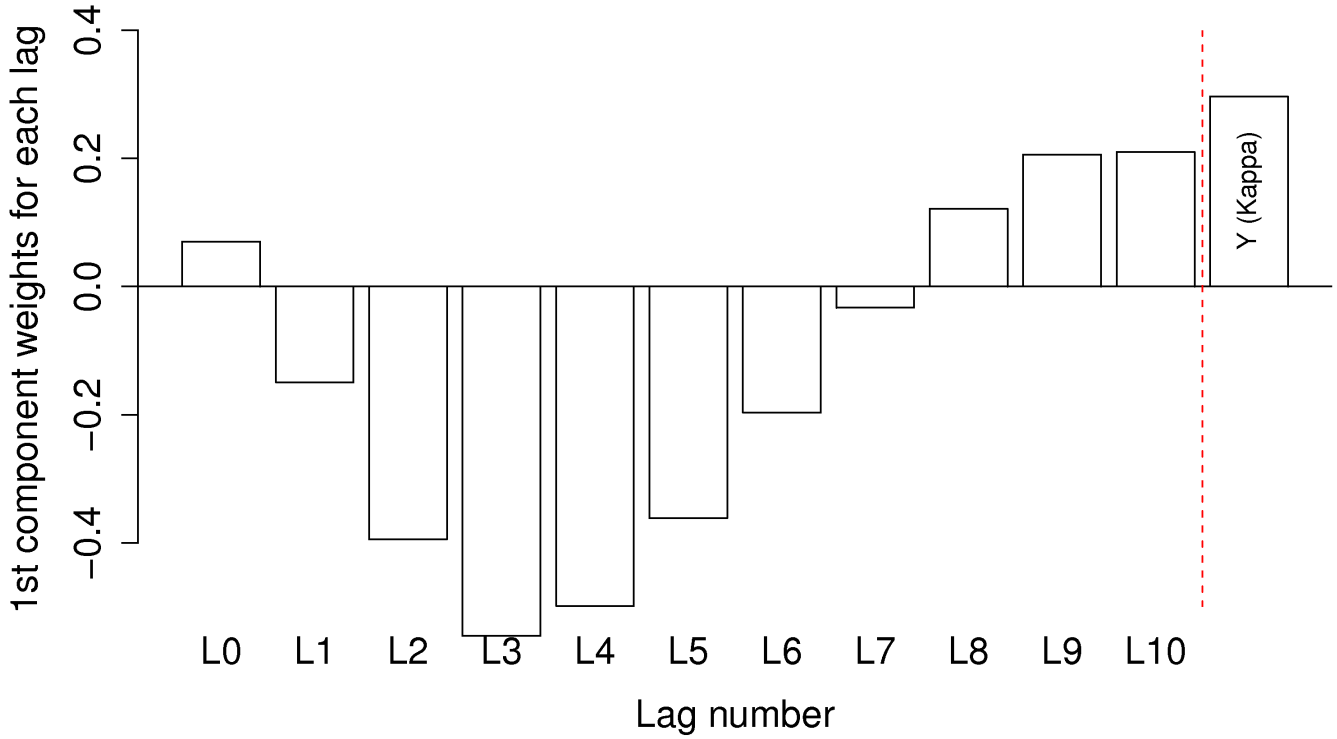


Time series example: lagging X's

- 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**

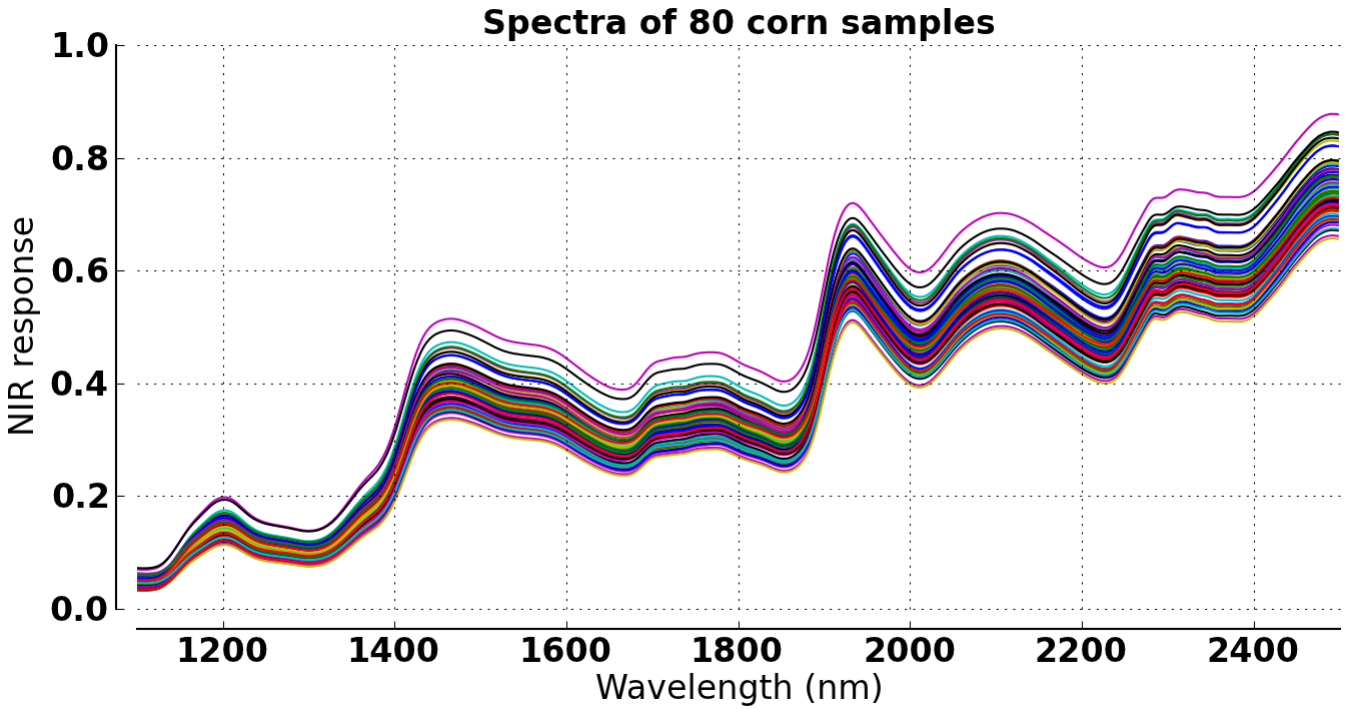
This can dominate a component, and cause false alarms for monitoring. Can't remove throughput variable, because it helps the model. Options?

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.

Spectral data

Spectral data have many specific preprocessing tools:

- Savitzky-Golay filtering/smoothing
- Multiplicative Scatter Correction (MSC)
- Standard Normal Variate (SNV) transformation
- Taking first derivatives wrt wavelength axis



Interesting project topic for someone working with spectra

Feature extraction: FFT and wavelet features

- Take FFT and/or wavelet features time-based signals
- Use these features in $\mathbf{X}$, instead of time-domain signals

Some features to consider:

- Amplitude in the time-domain signal
- Duration of a certain time-base signal
- Mean height of FFT peaks
- Area under certain/all FFT peaks

Empirical models: outline for this section

1. Cautions when interpreting empirical models
2. Correlation *vs* causation
3. Designed experiments are the only way to break the problem

Models

Famous quote

"Remember that all models are wrong; the practical question is how wrong do they have to be to not be useful."

George Box and Norman Draper, *Empirical Model-Building and Response Surfaces*, 1987, p 74.

- Models are useful:
  1. we can learn more about a process
  2. helpful for troubleshooting (model "before" and "after")
  3. make predictions from them
  4. used for monitoring
  5. sometimes used to optimize a process, especially cause-and-effect models

Models

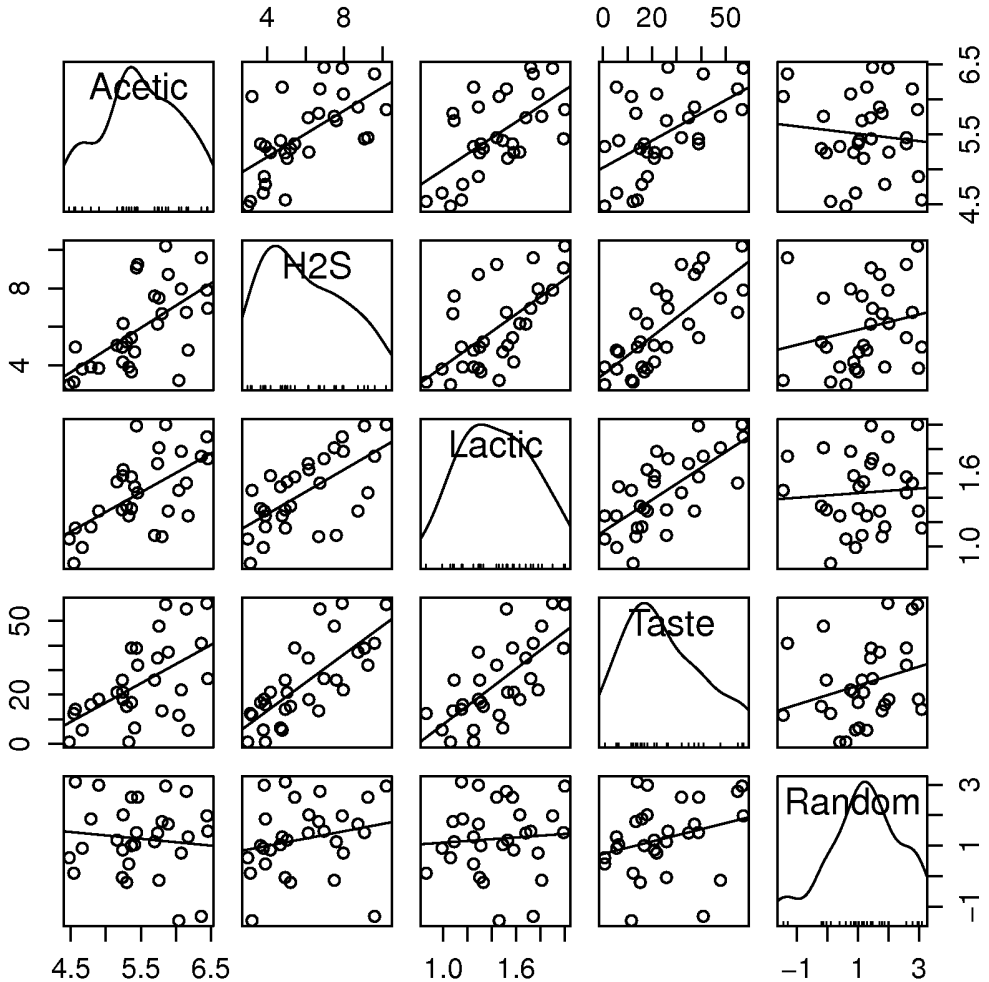
- Empirical model's **trends** often match reality
  - though exact values might not always be equal
- High fidelity first-principles models used with greater certainty
  - but are more costly to develop than empirical models

Aim

This section looks at the risks of using empirical models

Empirical model interpretation

By example: slightly modified cheese data set



Correlation matrix:

	Acetic	H2S	Lactic	Taste	Random
Acetic	1.000	0.618	0.6043	0.550	-0.1080
H2S	0.618	1.000	0.6439	0.756	0.2172
Lactic	0.604	0.644	1.0000	0.703	0.0735
Taste	0.550	0.756	0.7035	1.000	0.2790
Random	-0.108	0.217	0.0735	0.279	1.0000

- 3+1 X-variables
- We added a random variable
- $y =$ Taste
- $N = 30$

PCA model on the first 3 X-variables: how many components expected?

Some models for these data

Multiple linear regression

$$\widehat{y} = -38 + \underbrace{2.2x_{\text{acetic}}}_{-7.3 \le \beta_{\text{acetic}} \le 12} + \underbrace{3.4x_{\text{H2S}}}_{1.9 \le \beta_{\text{H2S}} \le 6.0} + \underbrace{19x_{\text{lactic}}}_{1.9 \le \beta_{\text{lactic}} \le 37} + \underbrace{2.3x_{\text{rand}}}_{-1.2 \le \beta_{\text{rand}} \le 5.8}$$

- RMSEE $= 9.4$ and $R^2 = 0.65$
- $\text{RMSEP}_{TSS} = 11.7 \longleftarrow$ test set switch $= 0.5(11.4 + 11.9)$

Variable selection: picks H2S

$$\widehat{y} = -9.8 + \underbrace{5.8x_{\text{H2S}}}_{3.8 \le \beta_{\text{H2S}} \le 7.7}$$

- RMSEE $= 10.5$ and $R^2 = 0.57$
- $\text{RMSEP}_{TSS} = 11.0$

Principal components regression:

$$\widehat{y} = \text{const} + 8.4x_{\text{acetic}} + 2.4x_{\text{H2S}} + 16x_{\text{lactic}} + 0.6x_{\text{rand}}$$

- RMSEE $= 9.9$ and $R^2 = 0.62$
- $\text{RMSEP}_{TSS} = 10.6$
- PCA loading $= \mathbf{p}_1 = [0.56, 0.59, 0.58, 0.09]$, explains 72.7%

Some models for these data

Projection to latent structures:

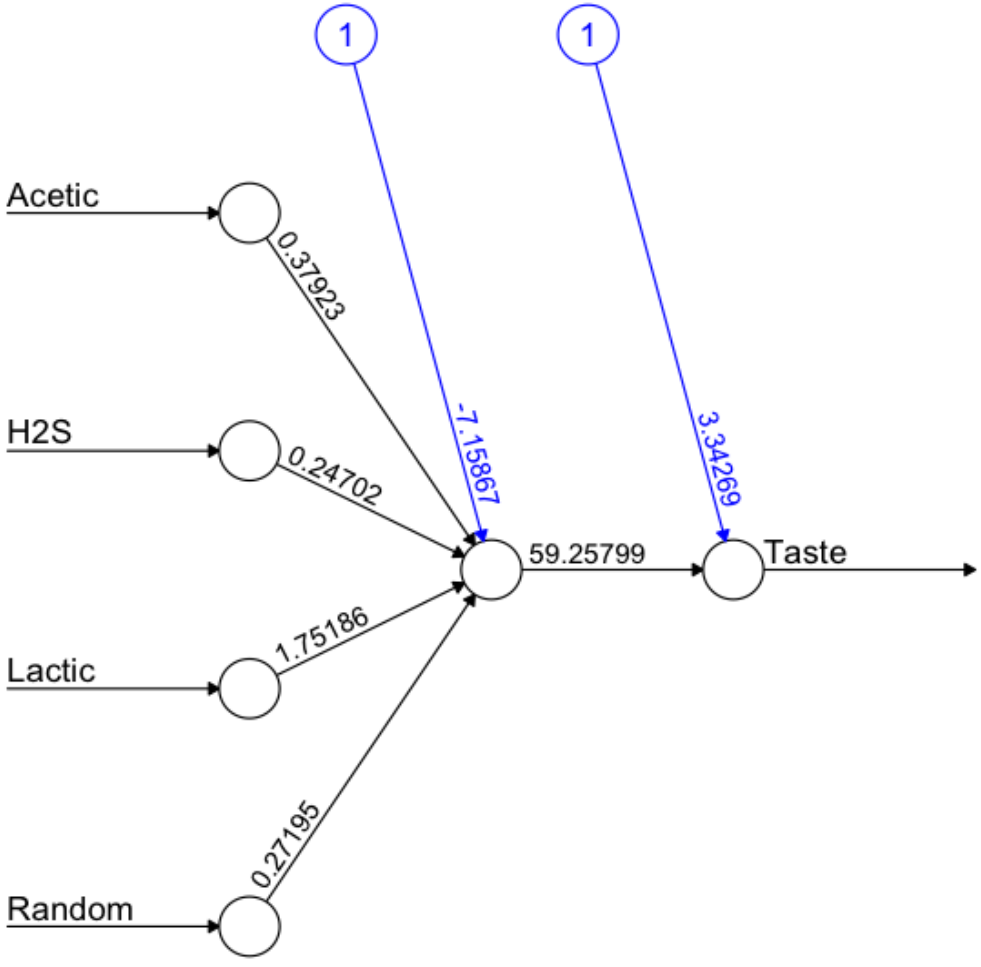
$\widehat{y} = \text{const} + 9.8x_{\text{acetic}} + 2.4x_{\text{H2S}} + 13.3x_{\text{lactic}} + 1.8x_{\text{rand}}$

- RMSEE $= 9.4$ and $R^2 = 0.65$
- $\text{RMSEP}_{TSS} = 10.7$
- PLS weights $= \mathbf{w}_1 = [0.46, 0.63, 0.59, 0.23]$

Neural network model: (also see next slide)

- RMSEE $= 8.8$ and $R^2 = 0.70$
- $\text{RMSEP}_{TSS} = 14.6$
- Weights: as shown on next page
- Confidence intervals for weights: all spanned zero

Neural network models



Error: 1166.392458 Steps: 16725

Trained in R. I will post the R script on the course website; feel free to make a better neural net model - my knowledge about them is limited.

Which model is appropriate?

For the purposes of

1. learning from the data?
2. troubleshooting?
3. making predictions?
4. process monitoring?
5. optimization?

We will discuss each of the above models in the context of these 5 purposes.

Process optimization

What if we wanted to improve taste (higher value)? Product development specialist suggests:

- high lactic acid: 9.0
- low H_2S: 4.0
- high acetic acid: 1.80
- random = 0

Model predictions are:

- MLR: 29.9
- OLS: 13.3
- PCR: 54.3
- PLS: 48.2
- NN: 38.8
- Which model is correct?
- None of them are correct! The correlation structure is broken.
    - SPE from PCR and PLS will be well above the limits

Inferring causality

- Only way to infer causality: designed experiment
  - Create $2^3 = 8$ new cheeses
  - Rebuild model
- Use your process knowledge to help infer causality
  - Some empirical models are causal: e.g. spectra related to corn
  - Digester data set we looked at earlier

Inferring causality

- Empirical models: only look at the correlation structure
- If correlation structure changes: rebuild the model
- Collinearity between variables:
  - PLS will spread the weight over all variables
  - MLR: cannot always invert $\mathbf{X}'\mathbf{X}$ with extreme collinearity

Noise in variables

- All models place less weight on noisy variables