Detecting anomalous data cells

Peter J. Rousseeuw
Wannes Van den Bossche

ROBUST@Leuven

61^{st} RBras, Salvador, May 24, 2016

Contamination types

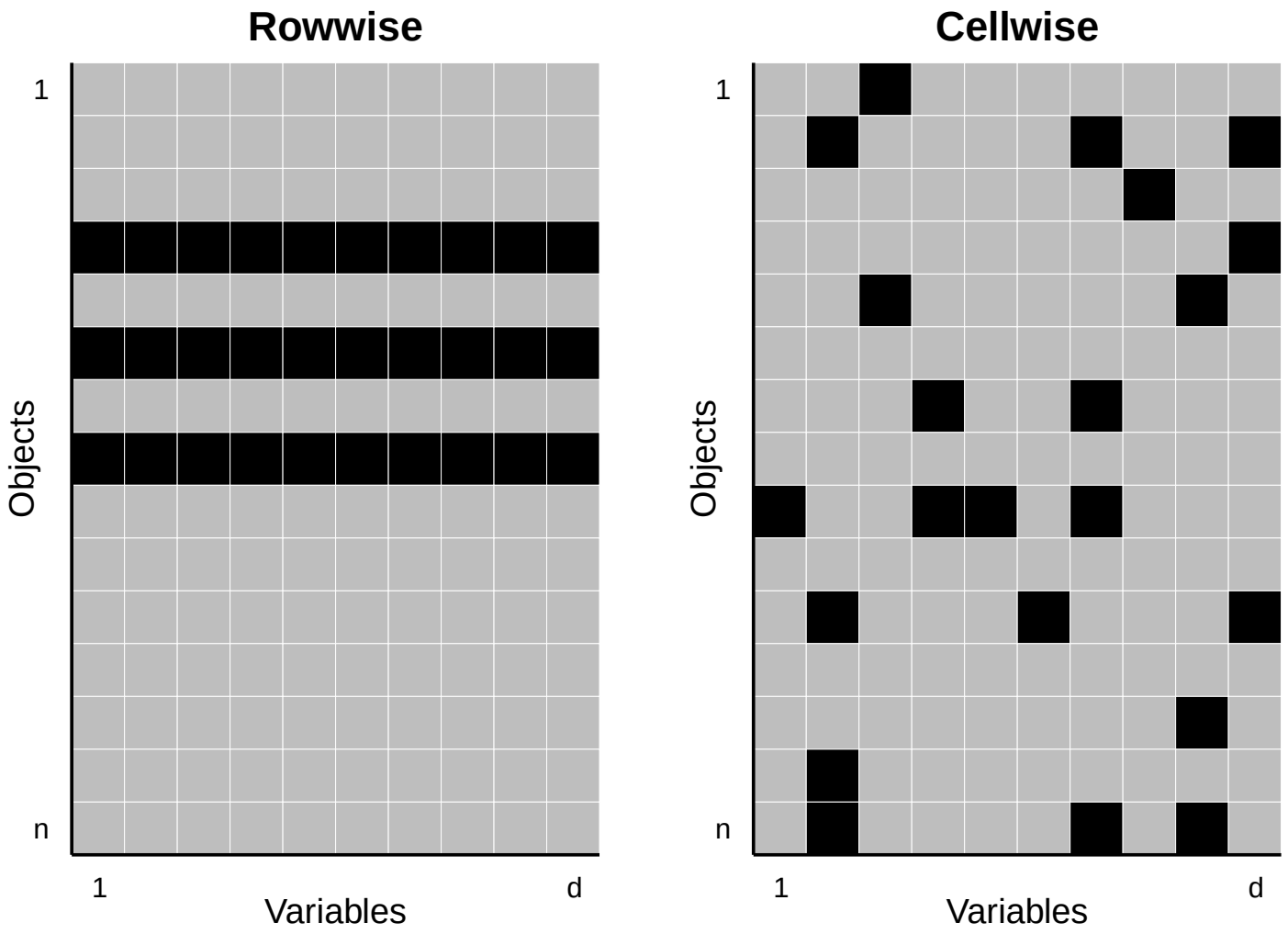
We want to analyze a data matrix $\boldsymbol{X}$ with n rows (cases) and $d > 1$ columns (variables). But the data may contain outliers.

The usual **rowwise contamination model** of Tukey (1960), Huber (1964),... assumes that some **rows** $\boldsymbol{x}_i$ have been replaced by arbitrary rows.

Such outlying rows may be cases belonging to a different population.
Many robust and equivariant estimators of covariance, regression, PCA,... were devised for this situation. They downweight or remove outlying rows.
These methods all require at least 50% of clean rows.

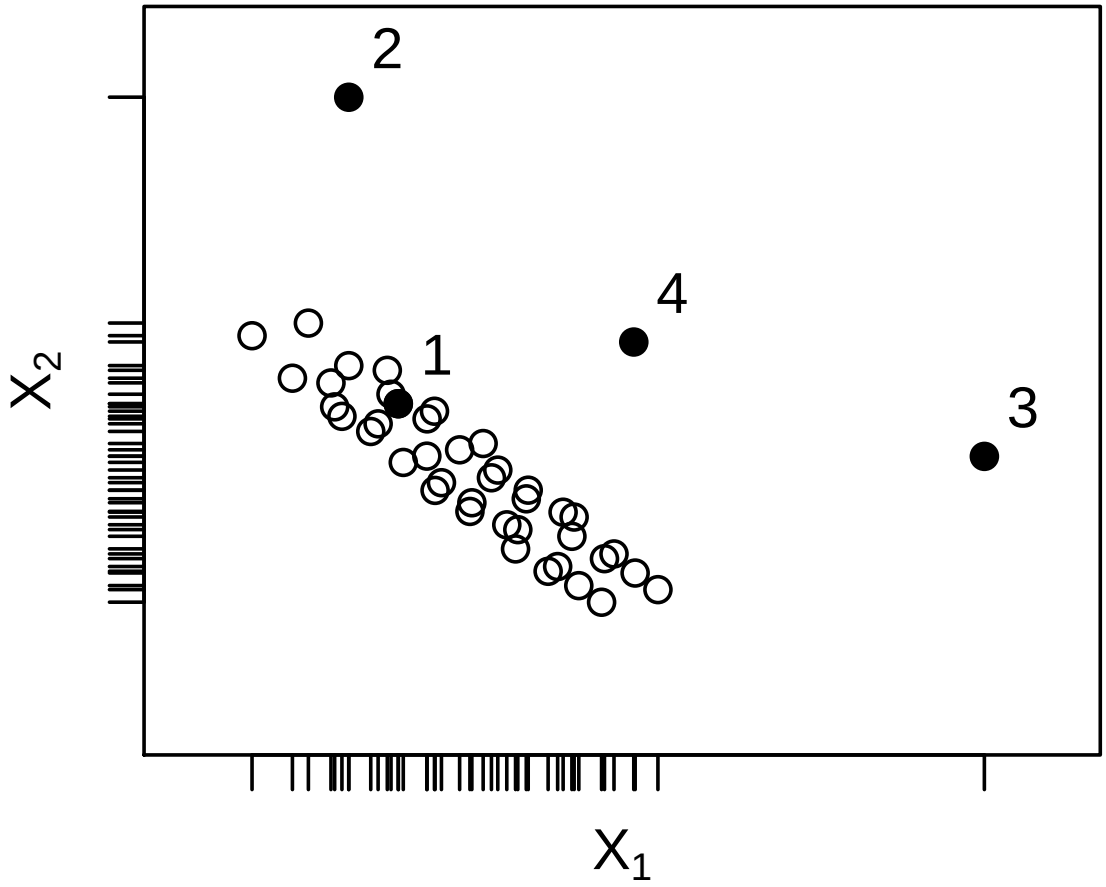
The **cellwise contamination model** of Alqallaf, Van Aelst, Yohai and Zamar (2009 AOS) assumes that some **cells** x_{ij} have been replaced.

In that case downweighting/deleting an entire row loses a lot of information.
For high d it is even possible that *every* row contains an outlying cell!



But how can we **detect** the outlying cells? Agostinelli, Leung, Yohai and Zamar (2015 Test) consider each column (variable) separately.

Bivariate example



Which cell is outlying: $x_{4,1}$ or $x_{4,2}$?

Algorithm **DetectDeviatingCells**

The R-code is available from *http://wis.kuleuven.be/stat/robust/software* and will be followed by a Matlab implementation.

Step 0: preprocessing. Check that the variables are roughly continuous, and temporarily set aside dummy variables.

It is okay when the data contain some missing values, but also set aside columns and rows with over 25% of NA's.

Verify that each remaining variable is approximately gaussian in its center, e.g. with QQ plots. If not, it is recommended to transform that variable so that the bulk of the data becomes roughly gaussian, e.g. by a robust version of the Box-Cox or Yeo-Johnson transformation.

Step 1: standardization. For each column j of $\boldsymbol{X}$ we estimate

$$m_j = \text{robLoc}_i(x_{ij}) \quad \text{and} \quad s_j = \text{robScale}_i(x_{ij} - m_j)$$

where *robLoc* is a robust estimator of location (such as the sample median) and *robScale* is a robust estimator of scale about zero.
Next, we standardize $\boldsymbol{X}$ to $\boldsymbol{Z}$ by $z_{ij} = (x_{ij} - m_j)/s_j$.

Step 2: univariate outlier detection. We define a new matrix $\boldsymbol{U}$ with entries $u_{ij} = z_{ij}$ except when

$$|z_{ij}| > c$$

in which case we set $u_{ij} =$ NA (missing). The cutoff value c is taken as

$$c = \sqrt{\chi^2_{1,p}}$$

where the probability p is 99% by default.

Step 3: bivariate relations. For any two columns $h \neq j$ we compute

$$\text{cor}_{jh} = \text{robCorr}_i(u_{ij}, u_{ih})$$

where **robCorr** is a robust correlation measure.

We only use the relation between variables j and h when

$$|\text{cor}_{jh}| \geqslant \text{corrlim}$$

in which corrlim $= 0.5$ by default. Variables j satisfying this for some $h \neq j$ will be called **connected**. The others are called **standalone** variables.

For the pairs (j, h) with $|\text{cor}_{jh}| \geqslant \text{corrlim}$ we also compute

$$b_{jh} = \text{robSlope}_i(u_{ij}|u_{ih})$$

where **robSlope** computes the slope of a robust no-intercept regression line that predicts variable j from variable h.

Step 4: estimated values. Next we compute estimated values $\hat{z}_{ij}$ for all cells. For each variable j we consider the set H_j consisting of all variables h with $|\text{cor}_{jh}| \geqslant \text{corrlim}$, including j itself. For all $i = 1, \ldots, n$ we then set

$$\hat{z}_{ij} = \frac{\sum_h w_{jh}\, b_{jh}\, u_{ih}}{\sum_h w_{jh}}$$

where $w_{jh} = |cor_{jh}|$. Other choices are possible, such as a weighted median.

Step 5: deshrinkage. Estimating $\hat{z}_{ij}$ often shrinks the scale of the entries, which is undesirable. To counteract the shrinkage we replace $\hat{z}_{ij}$ by

$$\hat{z}_{ij}\ \text{robSlope}_{i'}(z_{i'j}|\hat{z}_{i'j})$$

for all connected variables j.

Step 6: flagging cellwise outliers. Compute the standardized cell residuals

$$r_{ij} = \frac{z_{ij} - \hat{z}_{ij}}{\text{robScale}_{i'}\left(z_{i'j} - \hat{z}_{i'j}\right)} \quad .$$

In each column j we then flag all cells with $|r_{ij}| > c$ as cellwise outliers.

Next we assemble the 'imputed' matrix $\boldsymbol{Z}_{imp}$ given by

$$(\boldsymbol{Z}_{imp})_{ij} = \begin{cases} \hat{z}_{ij} & \text{if } z_{ij} \text{ was flagged or NA} \\ z_{ij} & \text{otherwise.} \end{cases}$$

Step 7: flagging rowwise outliers. The method can also flag some outlying rows i based on the standardized cell residuals r_{ij} .
For multivariate gaussian data without outliers we have $r_{ij} \approx N(0,1)$ so the cdf of r_{ij}^2 is approximately the cdf F of χ_1^2. This leads us to the criterion

$$T_i = \operatorname*{ave}_{j=1}^{d} F(r_{ij}^2) \quad .$$

We then robustly standardize the T_i and flag the rows i for which the standardized T_i exceeds the cutoff $\sqrt{\chi^2_{1,p}}$.

Note that when row i has an unusually large T_i this doesn't 'prove' that i is a member of a different population, but at least it is worth looking into.

Also note that although the T_i can flag **some** rowwise outliers, there are types of rowwise outliers that it may not detect. Therefore, it is recommended to use **rowwise robust** methods in subsequent analyses of the data.

Step 8: unstandardize. Turn the imputed matrix $\boldsymbol{Z}_{imp}$ into an imputed matrix $\boldsymbol{X}_{imp}$ by undoing the standardization. The output of DetectDeviatingCells is $\boldsymbol{X}_{imp}$ together with the list of cells and rows flagged as outlying.

DetectDeviatingCells does **not** require over 50% of clean rows!
It is equivariant for translations, for diagonal linear transformations, and for permuting rows and columns, but not for general linear transformations.

As a byproduct, DetectDeviatingCells **imputes** all NA's in the data.
This is far less efficient than the EM algorithm when the data are outlier-free, but it is more robust against cellwise outliers.

Note that DetectDeviatingCells starts from relations between 2 variables.
If instead we would fit a q-dimensional model to each set of $q > 2$ variables by a rowwise robust method, the computation time would explode and those fits would be less robust due to cellwise outlier propagation. The current algorithm runs in $O(nd^2)$ time and $O(nd)$ space, and speedups are possible.

Example: library(robustHD); data(TopGear)

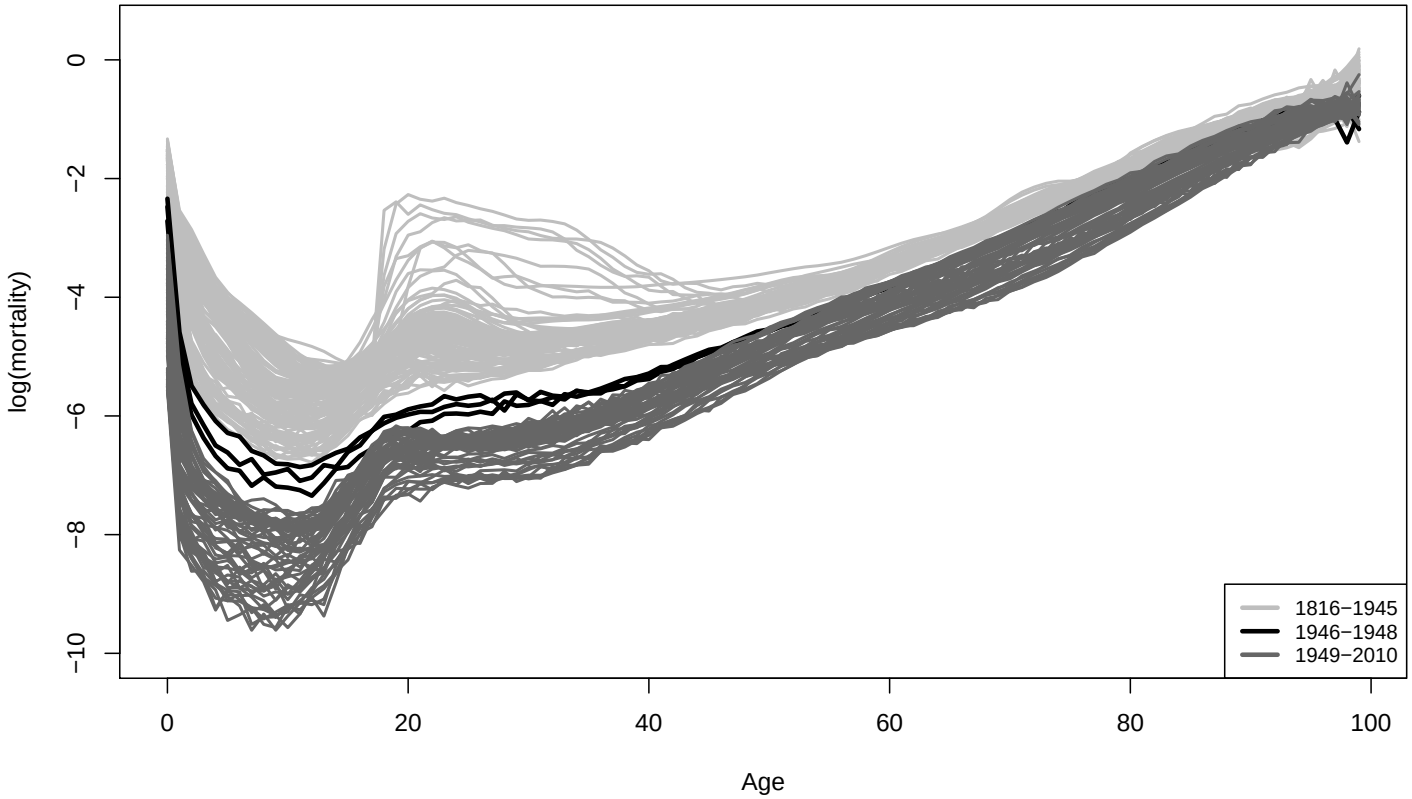
By column

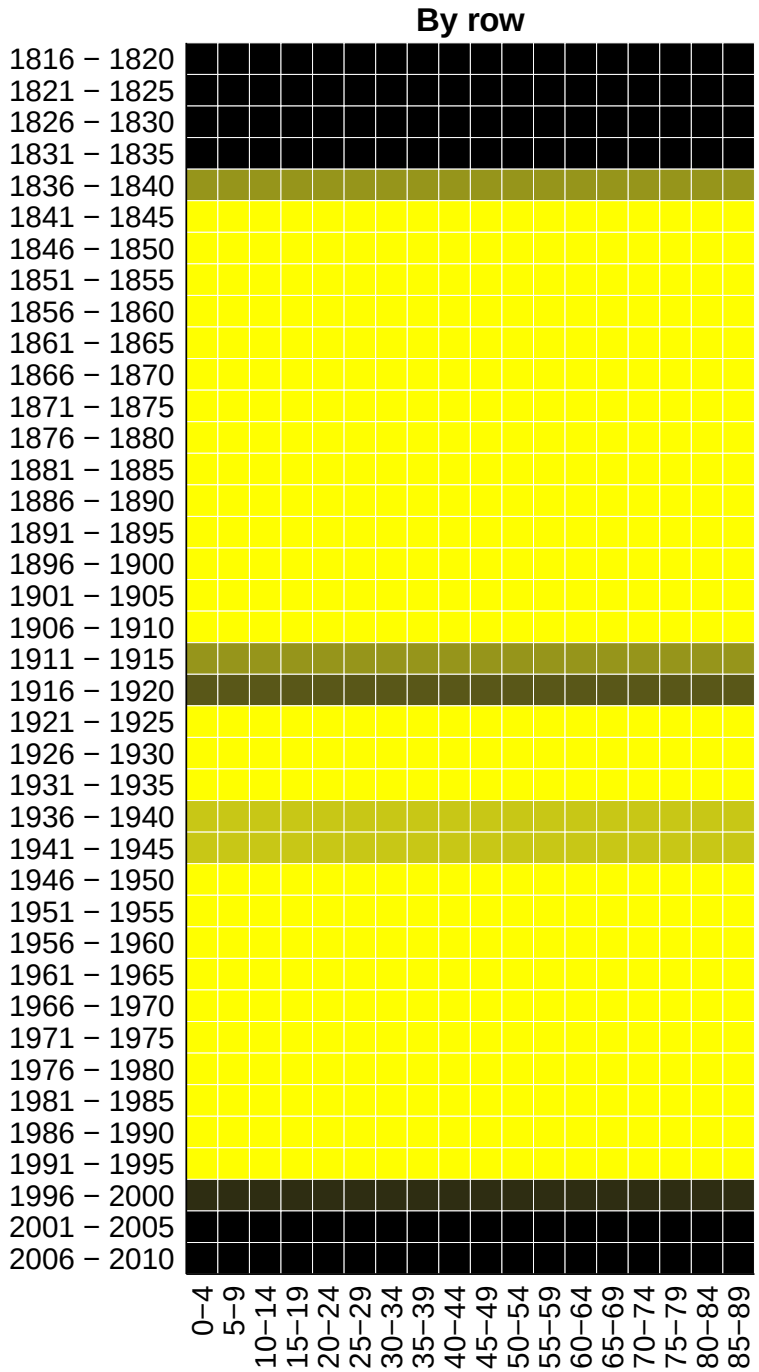
Price	Displacement	BHP	Torque	Acceleration	TopSpeed	MPG	Weight	Length	Width	Height	
31.5	1968	177	280	8.2	143	61	1480	4701	1826	1477	Audi A4
33.8	647	170	184	7.9	93	470	1315	3999	1775	1578	BMW i3
16.7	1991	125	221	10.3	122	51	1427	4597	1788	1477	Chevrolet Cruze
68.5	6162	437	424	4.3	186	21	1460	4435	1844	1246	Corvette C6
17.7	1368	160	169	7.4	131	43	1075	3660	1630	1490	Fiat 500 Abarth
26.8	1997	140	250	10.6	118	53	NA	4524	1838	1702	Ford Kuga
21.5	2199	150	258	8.5	135	67	1367	4300	1770	1470	Honda Civic
28.2	2198	122	265	14.7	90	25	2120	4785	1790	1790	Land Rover Defender
24.7	2191	150	280	9.4	122	54	1605	4555	1840	1710	Mazda CX−5
82.9	2987	211	398	9.1	108	25	2500	4662	1760	1951	Mercedes−Benz G
19.8	1598	184	192	6.9	143	48	NA	3734	1683	1378	Mini Coupe
9.3	998	68	70	14.2	100	65	210	3430	1630	1465	Peugeot 107
38.2	2706	265	206	5.8	164	34	1310	4374	1801	1282	Porsche Boxster
14.3	1461	90	162	12	112	88	1071	4062	1731	1448	Renault Clio
18	1998	155	265	0	105	38	2059	5125	1915	1845	Ssangyong Rodius
12	1328	84	81	14.1	87	39	1158	3665	1645	1705	Suzuki Jimny
20.1	1598	105	184	10.7	119	74	1295	4255	1799	1452	Volkswagen Golf

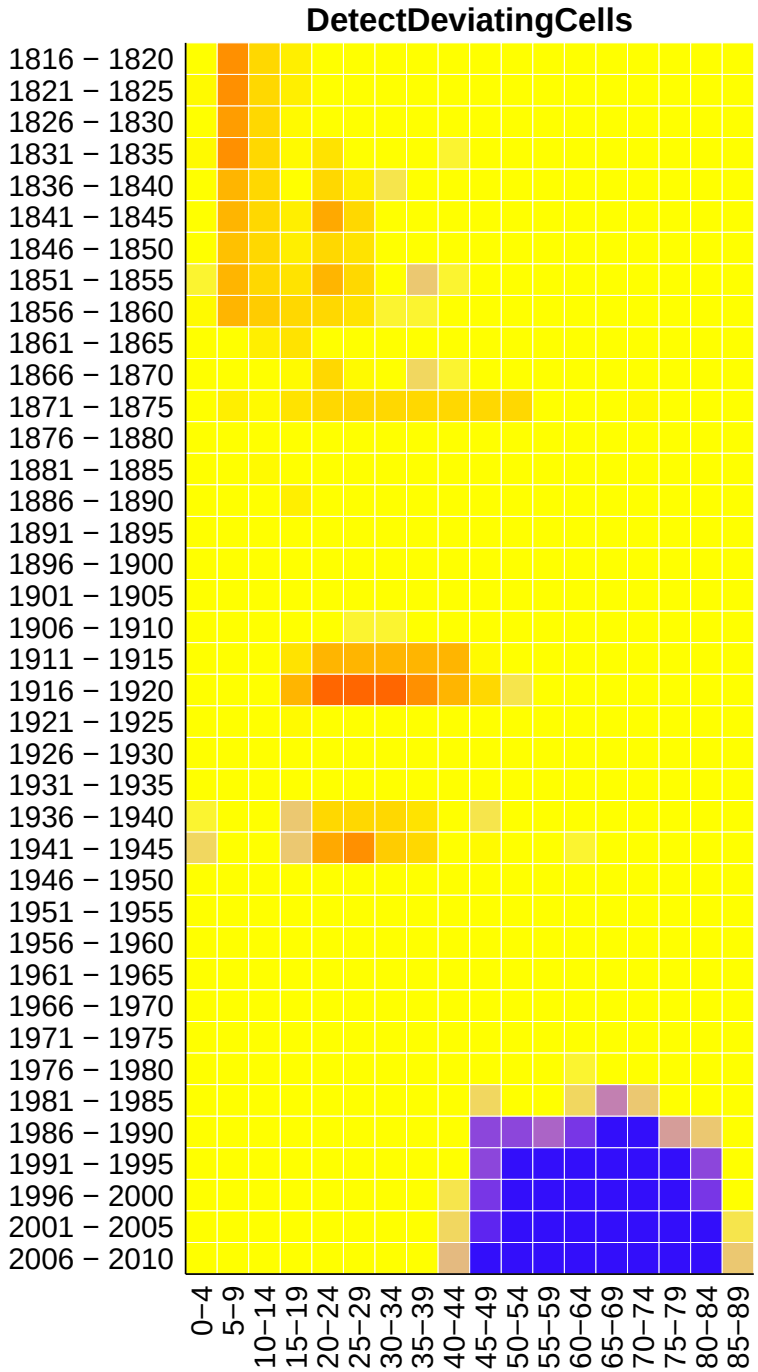
DetectDeviatingCells

Price	Displacement	BHP	Torque	Acceleration	TopSpeed	MPG	Weight	Length	Width	Height	
31.5	1968	177	280	8.2	143	61	1480	4701	1826	1477	Audi A4
33.8	647	170	184	7.9	93	470	1315	3999	1775	1578	BMW i3
16.7	1991	125	221	10.3	122	51	1427	4597	1788	1477	Chevrolet Cruze
68.5	6162	437	424	4.3	186	21	1460	4435	1844	1246	Corvette C6
17.7	1368	160	169	7.4	131	43	1075	3660	1630	1490	Fiat 500 Abarth
26.8	1997	140	250	10.6	118	53	NA	4524	1838	1702	Ford Kuga
21.5	2199	150	258	8.5	135	67	1367	4300	1770	1470	Honda Civic
28.2	2198	122	265	14.7	90	25	2120	4785	1790	1790	Land Rover Defender
24.7	2191	150	280	9.4	122	54	1605	4555	1840	1710	Mazda CX−5
82.9	2987	211	398	9.1	108	25	2500	4662	1760	1951	Mercedes−Benz G
19.8	1598	184	192	6.9	143	48	NA	3734	1683	1378	Mini Coupe
9.3	998	68	70	14.2	100	65	210	3430	1630	1465	Peugeot 107
38.2	2706	265	206	5.8	164	34	1310	4374	1801	1282	Porsche Boxster
14.3	1461	90	162	12	112	88	1071	4062	1731	1448	Renault Clio
18	1998	155	265	0	105	38	2059	5125	1915	1845	Ssangyong Rodius
12	1328	84	81	14.1	87	39	1158	3665	1645	1705	Suzuki Jimny
20.1	1598	105	184	10.7	119	74	1295	4255	1799	1452	Volkswagen Golf

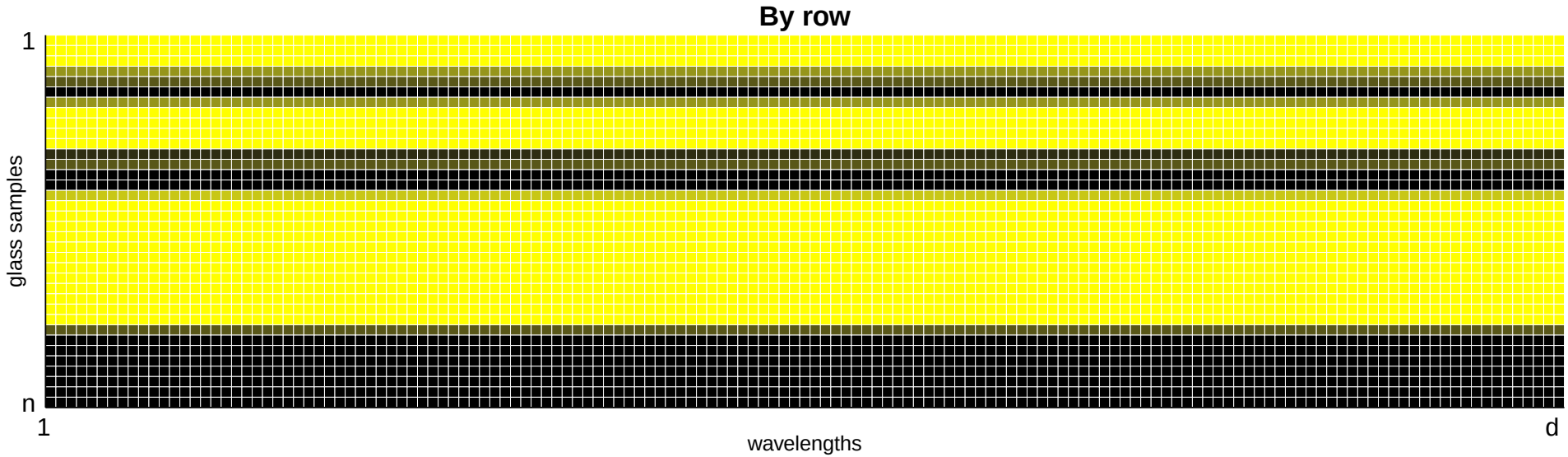
Mortality by age for males in France, from 1816 to 2010 (from www.mortality.org)

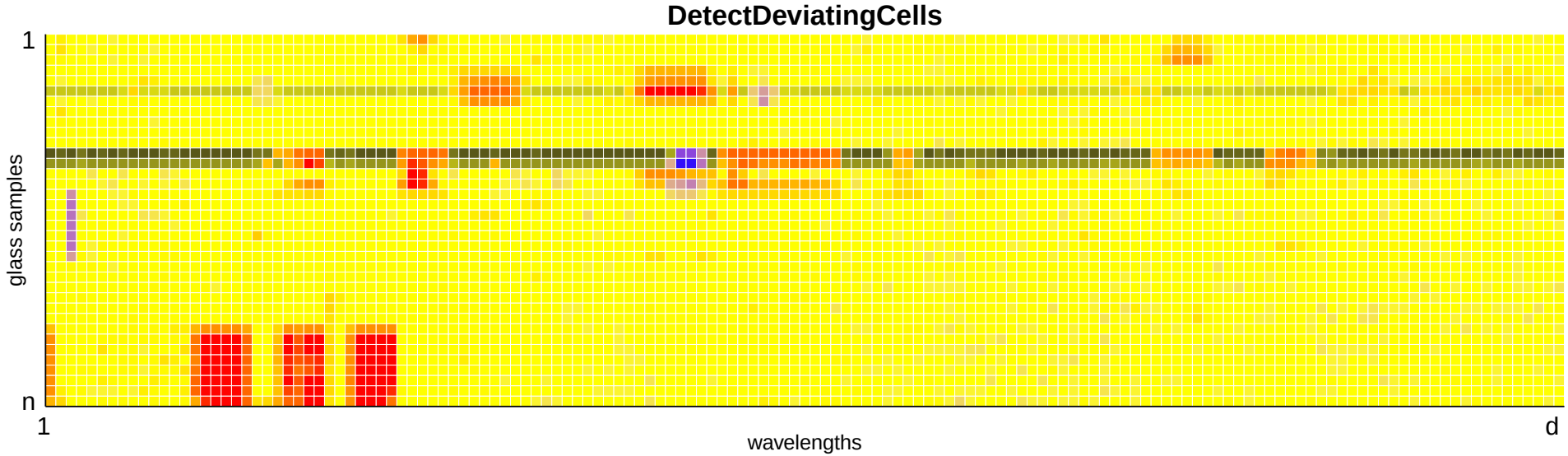



By row
1816 – 1820
1821 – 1825
1826 – 1830
1831 – 1835
1836 – 1840
1841 – 1845
1846 – 1850
1851 – 1855
1856 – 1860
1861 – 1865
1866 – 1870
1871 – 1875
1876 – 1880
1881 – 1885
1886 – 1890
1891 – 1895
1896 – 1900
1901 – 1905
1906 – 1910
1911 – 1915
1916 – 1920
1921 – 1925
1926 – 1930
1931 – 1935
1936 – 1940
1941 – 1945
1946 – 1950
1951 – 1955
1956 – 1960
1961 – 1965
1966 – 1970
1971 – 1975
1976 – 1980
1981 – 1985
1986 – 1990
1991 – 1995
1996 – 2000
2001 – 2005
2006 – 2010
0–4
5–9
10–14
15–19
20–24
25–29
30–34
35–39
40–44
45–49
50–54
55–59
60–64
65–69
70–74
75–79
80–84
85–89


DetectDeviatingCells
1816 – 1820
1821 – 1825
1826 – 1830
1831 – 1835
1836 – 1840
1841 – 1845
1846 – 1850
1851 – 1855
1856 – 1860
1861 – 1865
1866 – 1870
1871 – 1875
1876 – 1880
1881 – 1885
1886 – 1890
1891 – 1895
1896 – 1900
1901 – 1905
1906 – 1910
1911 – 1915
1916 – 1920
1921 – 1925
1926 – 1930
1931 – 1935
1936 – 1940
1941 – 1945
1946 – 1950
1951 – 1955
1956 – 1960
1961 – 1965
1966 – 1970
1971 – 1975
1976 – 1980
1981 – 1985
1986 – 1990
1991 – 1995
1996 – 2000
2001 – 2005
2006 – 2010
0–4
5–9
10–14
15–19
20–24
25–29
30–34
35–39
40–44
45–49
50–54
55–59
60–64
65–69
70–74
75–79
80–84
85–89

Example: n=180 archeological glass samples, spectra with d=750 wavelengths, so more dimensions than objects!





After running the algorithm DetectDeviatingCells:

1. Ideally, the user looks at the anomalous cells and whether their values are higher or lower than expected, and makes sense of what is going on. This may lead to a better understanding of the data pattern, to changes in the way the data are collected/measured, to deselecting certain rows or columns, to transforming variables, to changing the model,...

2. If the data set is too large for visual inspection of the results or the analysis is automated, the anomalous cells can be set to missing after which the data set is analyzed by a method appropriate for incomplete data.

3. If no such method is available, one can analyze the imputed data set $\boldsymbol{X}_{imp}$ produced by DetectDeviatingCells, which has no missings.

If 2. or 3. are carried out by a sparse method such as the Lasso (Tibshirani 1996, Städler-Stekhoven-Bühlmann 2014) or another form of variable selection: look more closely at the anomalous cells in the selected variables.

Comparison with existing method

We compare with the univariate **Gervini-Yohai (GY)** filter (2002 AOS).

The **uncontaminated** data are gaussian with mean zero and a covariance matrix with unit diagonal. We use two types of correlation matrices:

- **ALYZ**: the random correlation matrices generated by Agostinelli, Leung, Yohai and Zamar (2015 Test). These yield relatively low correlations.
- **A09**: the true correlation matrix is generated as

$$\rho_{jh} = (-0.9)^{|j-h|} \quad .$$

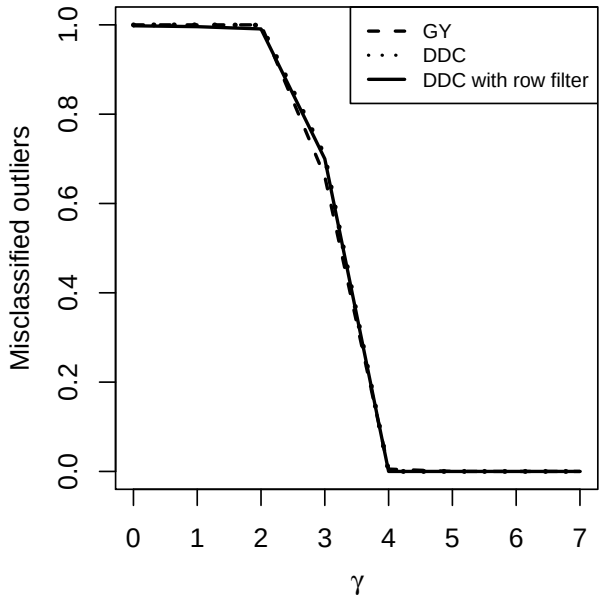
The two types of **contamination** are generated as follows:

- **cells**: A random subset of the nd cells are replaced by the constant γ.
- **rows**: compute the last eigenvector $\boldsymbol{v}$ of the true covariance matrix $\boldsymbol{C}$. Rescale $\boldsymbol{v}$ to the typical size of a data point, by making

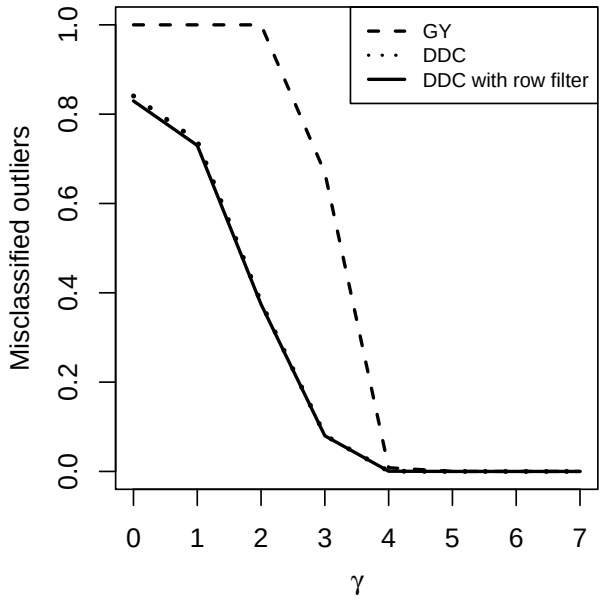
$$MD^2_{\boldsymbol{C}}(\boldsymbol{v}) = E[Y^2] = d \qquad \text{where} \qquad Y^2 \sim \chi^2_d \quad .$$

  Then add point contamination at $\gamma\boldsymbol{v}$.

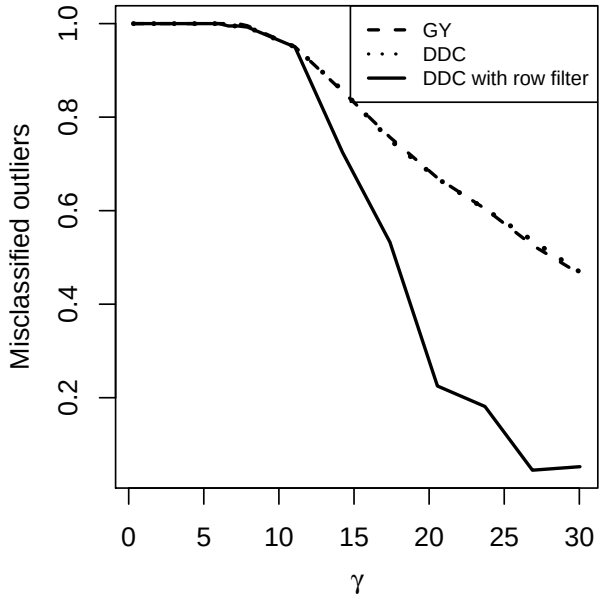
ALYZ model, 10% cells, d=20



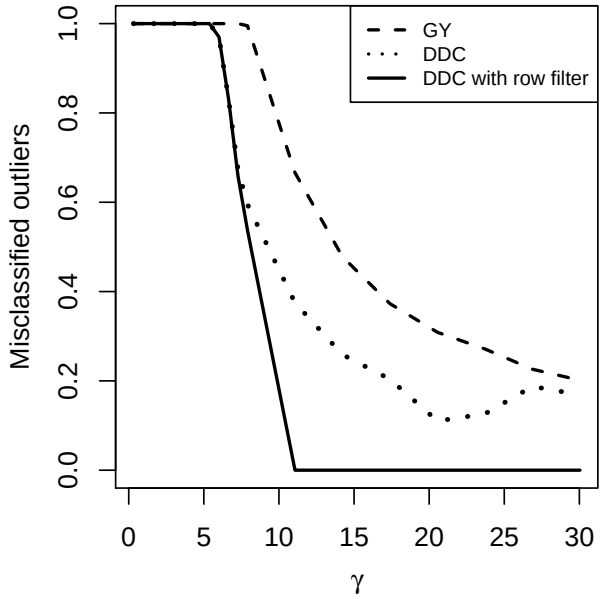
A09 model, 10% cells, d=20



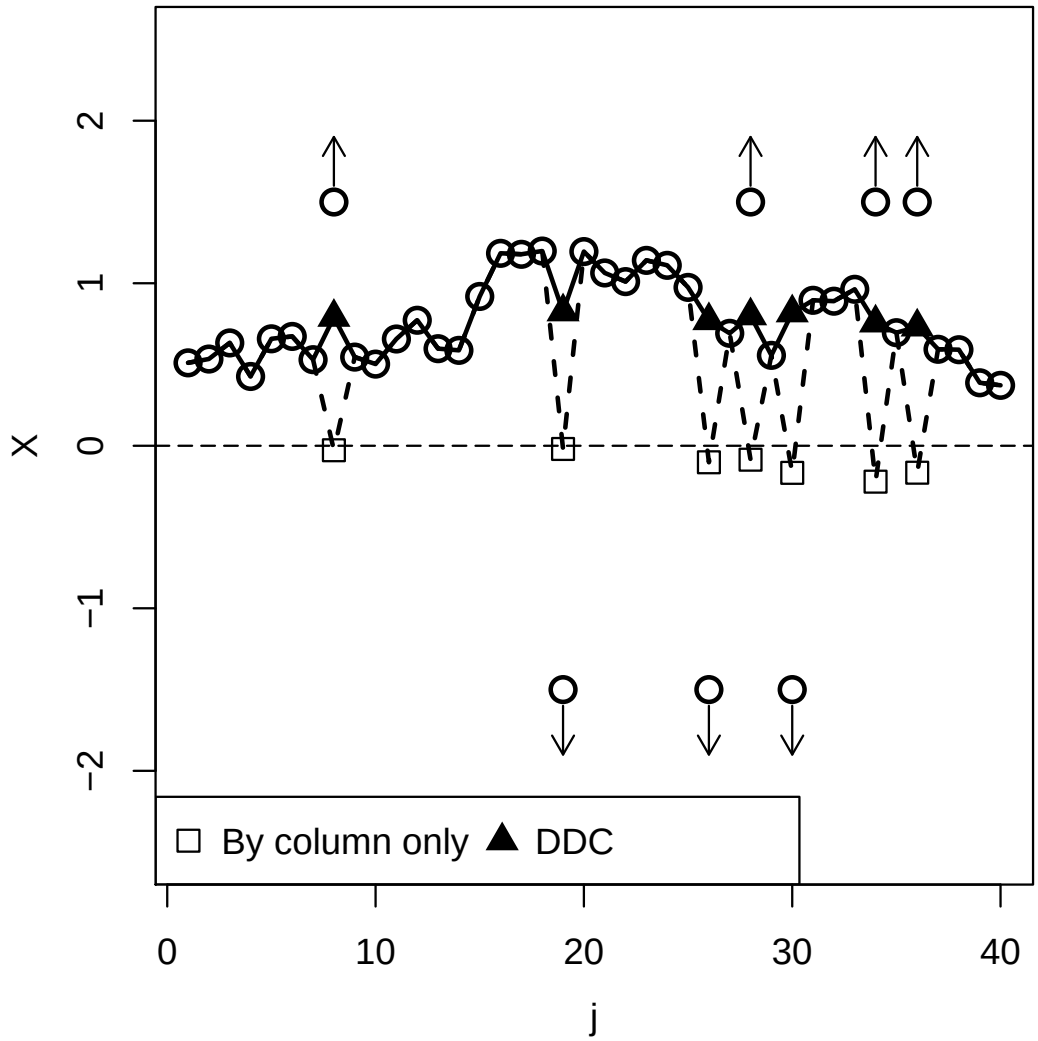
ALYZ model, 10% rows, d=20



A09 model, 10% rows, d=20



If the true correlations are given by $\rho_{jh} = 0.99^{|j-h|}$ then the rows of $\boldsymbol{X}$ look like autocorrelated time series.
Compare DetectDeviatingCells imputation to median imputation by column:



Multivariate location and scatter

The 2SGS method (Agostinelli, Leung, Yohai and Zamar 2015) consists of:

1. Apply the **GY** filter to the columns of $\boldsymbol{X}$ and set the flagged cells to NA
2. Apply the **GSE** estimator of Danilov, Yohai and Zamar (2012 JASA) to this incomplete data set, yielding $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$.

Our version replaces GY in step 1 by DetectDeviatingCells. When an entire row is flagged, we remove it.

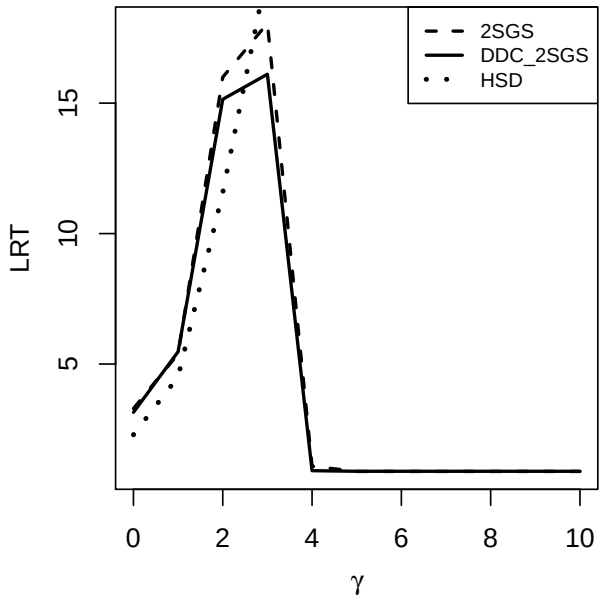
Removing a row will count as removing all of its d cells.

We also include the HSD estimator of Van Aelst et al (2012).

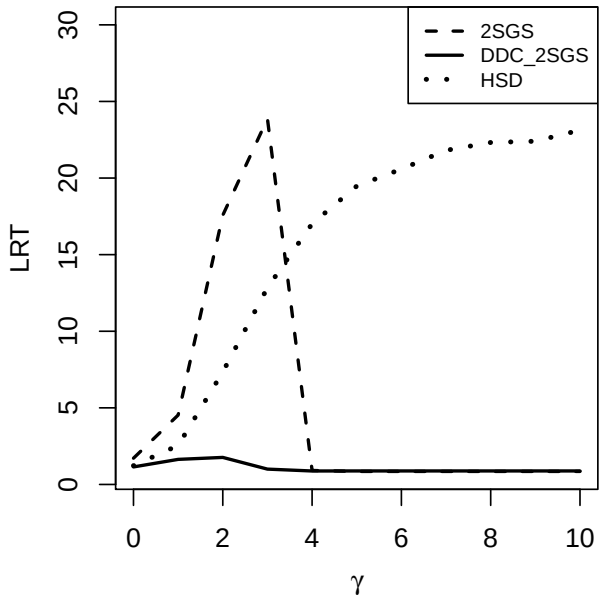
Measure how far $\hat{\boldsymbol{\Sigma}}$ is from the true $\boldsymbol{\Sigma}$ by

$$\text{LRT} = \text{trace}(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}^{-1}) - \log(\det(\hat{\boldsymbol{\Sigma}}\boldsymbol{\Sigma}^{-1}) - d \quad .$$

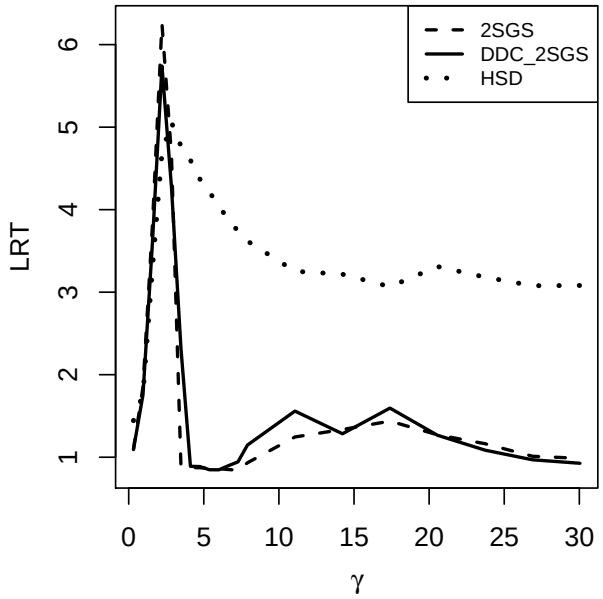
LRT for ALYZ, 10% cells, d=10



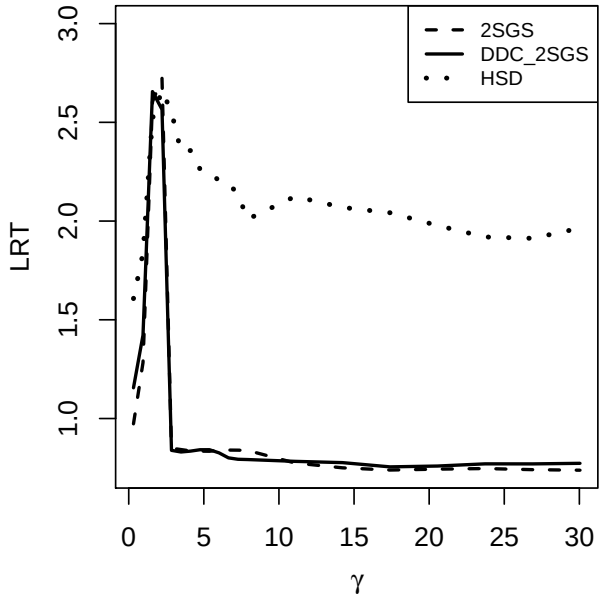
LRT for A09, 10% cells, d=10



LRT for ALYZ, 10% rows, d=10



LRT for A09, 10% rows, d=10



Three-step versions

Note that the default GSE estimator starts by computing a version of the minimum volume ellipsoid (MVE) suitable for incomplete data, which is rather time-consuming.

Therefore Agostinelli, Leung, Yohai and Zamar (Test 2015, rejoinder) also proposed **Fast2SGS**:

1. Apply the GY filter to the columns of $\boldsymbol{X}$ and set the flagged cells to NA.
2. Apply the MVE_S estimator (an S-estimator starting from MVE) to the complete data set obtained by imputing the NA's by the median of their column;
3. Apply GSE to the data set with the NA's but now starting from the $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ obtained in step 2.

Our version again replaces GY in step 1 by DetectDeviatingCells.

Another approach to estimating $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ is the **SnipEM** method of Farcomeni (2014 Technometrics). This searches for the subset of cells such that 'snipping' them (setting them to NA) and then running the EM algorithm on this incomplete data set yields the highest partial likelihood.

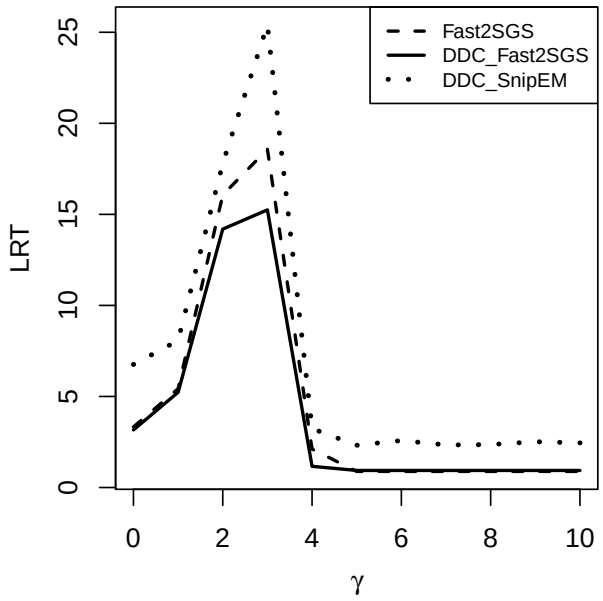
The SnipEM algorithm requires a good initial subset.

Agostinelli et al (2015) found that SnipEM works fairly well for cell contamination but is insufficiently robust for row contamination.

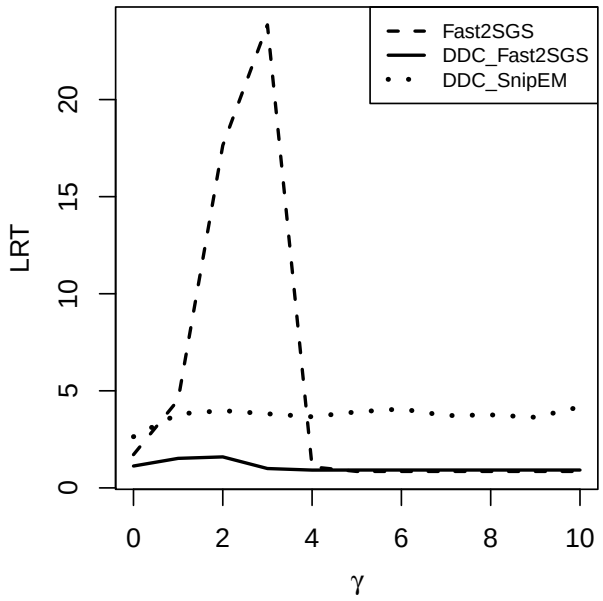
Therefore we take the following steps:

1. Apply DetectDeviatingCells to $\boldsymbol{X}$ with its imputation of the flagged cells and remove the flagged rows.
2. Apply the MVE_S estimator to this modified data set and flag the outlying rows it detects too.
3. Apply SnipEM to the data set without the rows flagged in steps 1 and 2. As initial subset we use the cells flagged in step 1.

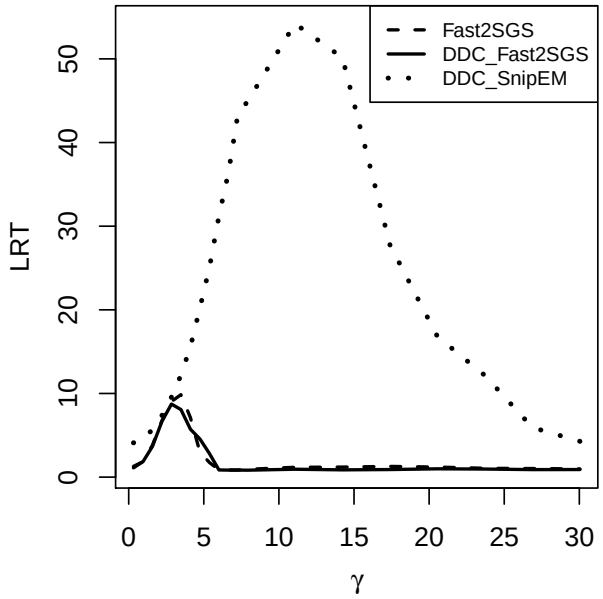
LRT for ALYZ, 10% cells, d=10



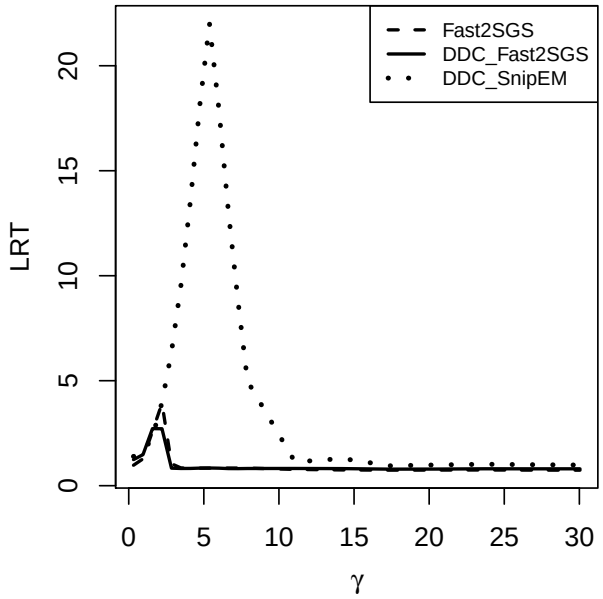
LRT for A09, 10% cells, d=10



LRT for ALYZ, 10% rows, d=10



LRT for A09, 10% rows, d=10



Regression

Leung, Zhang and Zamar (2015 arXiv) proposed the **3S** regression method:

1. Apply a generalization of the GY filter to the rows of $\boldsymbol{X}$ (not y) and set the flagged cells to NA
2. Apply GSE to the incomplete data set $[\boldsymbol{X}|y]$, yielding $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$
3. Partitioning $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ as

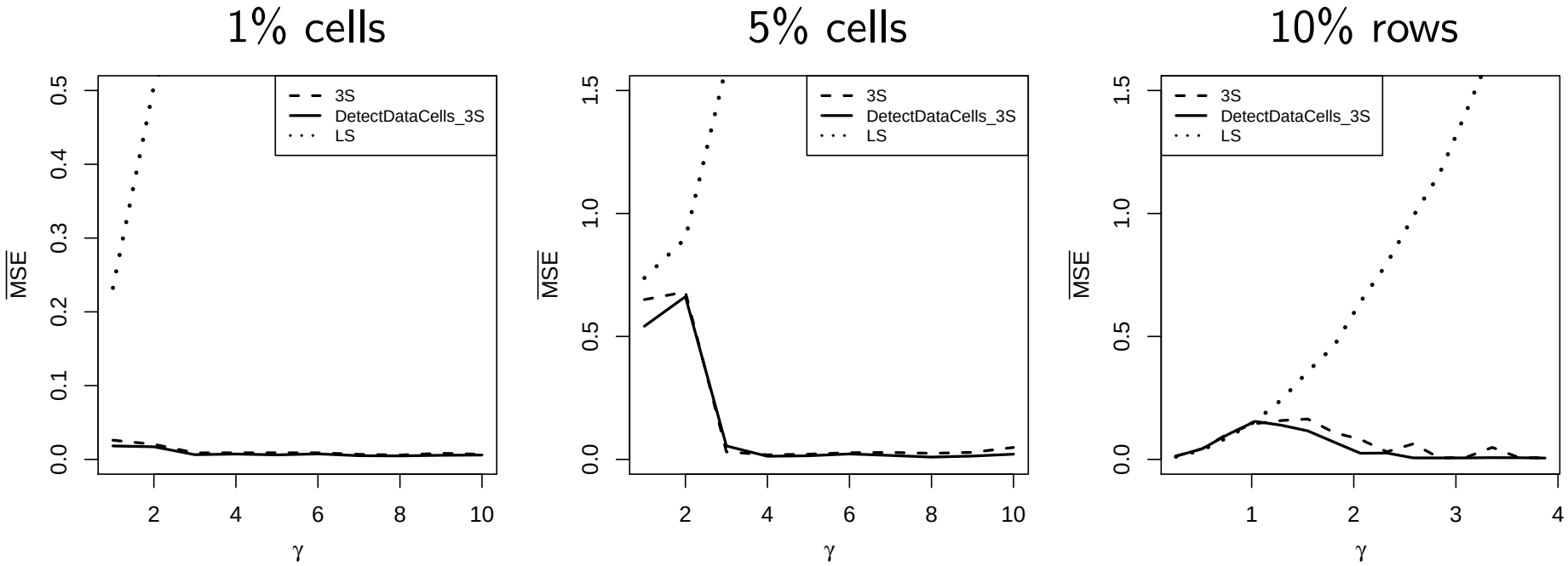
$$\hat{\boldsymbol{\mu}} = \begin{pmatrix} \hat{\boldsymbol{\mu}}_x \\ \hat{\boldsymbol{\mu}}_y \end{pmatrix} \quad \text{and} \quad \hat{\boldsymbol{\Sigma}} = \begin{pmatrix} \hat{\boldsymbol{\Sigma}}_{xx} & \hat{\boldsymbol{\Sigma}}_{xy} \\ \hat{\boldsymbol{\Sigma}}_{yx} & \hat{\boldsymbol{\Sigma}}_{yy} \end{pmatrix}$$

yields the slope and intercept estimates

$$\begin{aligned} \hat{\boldsymbol{\beta}} &= \hat{\boldsymbol{\Sigma}}_{xx}^{-1} \hat{\boldsymbol{\Sigma}}_{xy} \\ \hat{\boldsymbol{\alpha}} &= \hat{\boldsymbol{\mu}}_y - \hat{\boldsymbol{\beta}}^t \hat{\boldsymbol{\mu}}_x \quad . \end{aligned}$$

This is again robust to both cellwise and rowwise outliers.
Our version replaces step 1 by DetectDeviatingCells.

Average MSE$(\hat{\beta}, \hat{\alpha})$ under ALYZ with cell and row contamination (d=15, n=300):



Under ALYZ, the average lengths of the confidence intervals around the coefficients is similar to those found by Leung-Zhang-Zamar, for the same contamination settings.

The same holds for the average coverage rates of those intervals.

Conclusions and Outlook

Our approach turns high dimensionality (usually considered a **curse**) into an advantage, as having more variables may improve the accuracy of the estimated cells.

DetectDeviatingCells and methods starting from it perform about as well as the robust methods of Zamar et al (GY filter, 2SGS, Fast2SGS, 3S regression) when the correlations between the variables are small to moderate, and better when there are higher correlations.

Conclusions and Outlook

Our approach turns high dimensionality (usually considered a **curse**) into an advantage, as having more variables may improve the accuracy of the estimated cells.

DetectDeviatingCells and methods starting from it perform about as well as the robust methods of Zamar et al (GY filter, 2SGS, Fast2SGS, 3S regression) when the correlations between the variables are small to moderate, and better when there are higher correlations.

OBRIGADO!