Exercise 5

Business Analytics and Data Science WS16/17

Introduction

We will explore the problem of overfitting also described by the bias-variance-tradeoff by comparing standard logistic regression to two of its regularized forms: Lasso and Ridge regression. These models follow the credo that sometimes less is more by putting a penalty on the size of the coefficients. Intuitively, we force the model through regularization to fit the data it knows less well in the hope that it will be more useful when working on unknown data. In the same way, we hope that you will understand the general point of the exercise rather than memorize its specificities and details. For example, did you know that ridge regression is also called weight decay?

Logistic regression

Logistic regression is one of the standard models in predictive analytics, modeling the outcome of a categoric variable by a linear combination of independent variables transformed in a way to predict outcome probabilities between 0 and 1.

0. Load the *loans* dataset.
1. Run a logistic regression to predict the dependent variable **BAD** based on all other variables in the data set. Use the **glm()** generalized model function to create the model and save the trained model as **lr**.
   Hint 1: Check the help on **formula** on how to specify "all other variables".
   Hint 2: Specify that you need a logistic model with option **family**. Check the help for **family** to find out which specification to use if you are unsure.
2. Which variables have a significant impact on the value of **BAD**? In order to identify the significant variables, extract the *p-values* of model coefficients from the model object. Collect the column indices of all variables that are significant on the 5% level.
3. Use the trained model **lr** to predict for each observation in the **loans** data frame the probability that the applicant is a bad risk. Use function **predict()** and store the predictions in a variable **pred.lr**.
4. Evaluate the model by computing the prediction accuracy of the model. This is defined as the ratio of correct predictions (*good* if observed *good*, *bad* if observed *bad*) to the overall number of predictions.
5. Last, compute the Brier score **brier.lr** of your model to evaluate not only accuracy but also the *calibration* of your model. The Brier score is the mean square error (MSE) between the actual values of **BAD** (1/0) and the predicted class probabilities **pred.lr**. How well does your model perform?

```
##
## Call:
## glm(formula = BAD ~ ., family = binomial(link = "logit"), data = loans)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -1.6077  -0.7811  -0.6422   1.0098   2.1960
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept)  5.906e-01  6.814e-01   0.867   0.3861
## YOB         -1.586e-02  7.583e-03  -2.092   0.0364 *
## nKIDS       -3.873e-02  7.643e-02  -0.507   0.6123
## nDEP         2.554e-01  3.018e-01   0.846   0.3973
## PHON        -1.408e-01  2.327e-01  -0.605   0.5451
## dINC_SP     -4.531e-05  1.807e-05  -2.508   0.0122 *
## EMPS_AE     -2.042e-01  4.816e-01  -0.424   0.6716
## EMPS_AM     -2.302e-01  6.651e-01  -0.346   0.7293
## EMPS_AN     -7.721e-01  1.206e+00  -0.640   0.5219
## EMPS_AP     -3.607e-01  4.435e-01  -0.813   0.4160
## EMPS_AR     -2.411e-01  5.440e-01  -0.443   0.6577
## EMPS_AT     -3.441e-01  5.026e-01  -0.685   0.4935
## EMPS_AU      3.148e-01  8.609e-01   0.366   0.7146
## EMPS_AV     -5.291e-01  4.623e-01  -1.145   0.2524
## EMPS_AW      1.866e-01  5.929e-01   0.315   0.7530
## EMPS_AZ     -4.795e-01  9.706e-01  -0.494   0.6213
## dINC_A      -2.673e-05  5.951e-06  -4.491 7.09e-06 ***
## RESN         9.372e-01  3.534e-01   2.652   0.0080 **
## RESO         4.369e-01  3.115e-01   1.403   0.1607
## RESP         1.122e-01  2.739e-01   0.410   0.6820
## RESU        -2.606e-02  3.114e-01  -0.084   0.9333
## dHVAL       -4.920e-06  4.811e-06  -1.023   0.3065
## dMBO         2.848e-06  5.028e-06   0.566   0.5712
## dOUTM        2.256e-04  2.050e-04   1.101   0.2711
## dOUTL        7.909e-05  1.271e-04   0.623   0.5336
## dOUTHP      -1.296e-03  8.622e-04  -1.504   0.1327
## dOUTCC      -2.308e-03  9.491e-04  -2.431   0.0150 *
## YOB_missing  3.029e-01  9.900e-01   0.306   0.7596
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 1413.3  on 1224  degrees of freedom
## Residual deviance: 1308.3  on 1197  degrees of freedom
## AIC: 1364.3
##
## Number of Fisher Scoring iterations: 5
```

```
##          1          2          3          4          5          6
##  0.1180762 -1.2330187 -1.2340800 -0.4529083 -1.2308004 -1.6483642
```

```
## [1] 0 0 1 0 1 0
```

```
## [1] "Logistic regression has a Brier Score of 0.17784 (lower is better)"
```

```
## [1] "Reduced logistic regression model has a Brier Score of 0.18147 (c.f. 0.17784 of the model with a
```

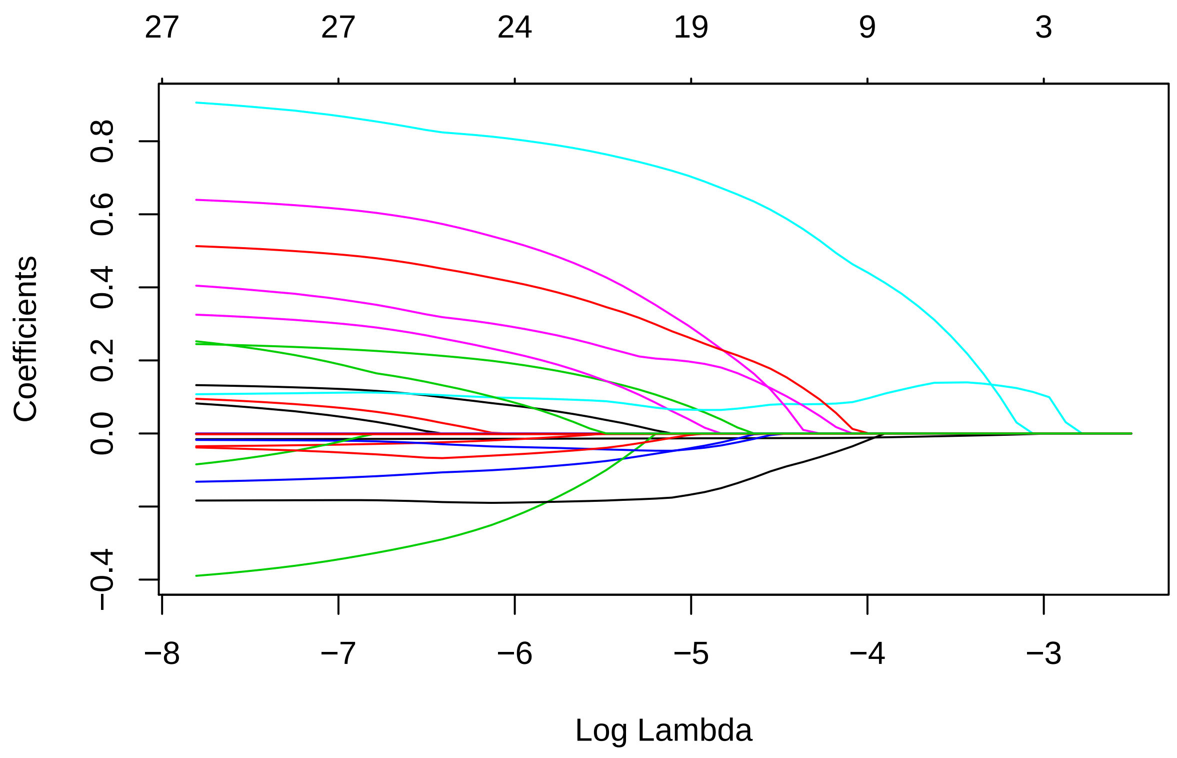
Regularization

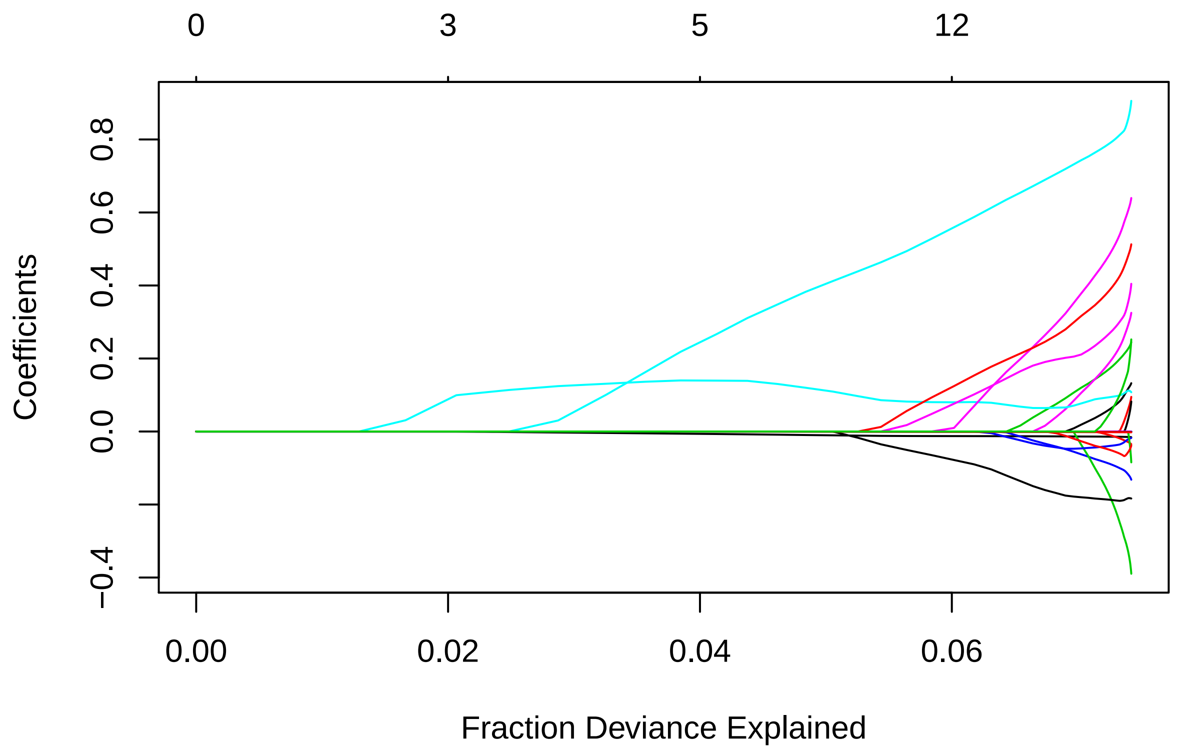
We have discussed two forms of common regularization for the example of logistic regression. Both work by including a measure of coefficient size into the loss function, i.e. the function which the algorithm optimizes, in the form of a penalty. Intuitively, instead of telling the algorithm to build a model that fits well, we now tell it to build a model that fits well *and* keeps the coefficients small by deducting points in relation to the size of coefficients.

The difference between the *lasso* and *ridge* penalty is then only whether we substract the absolute or the squared sum of coefficients. While lasso tends to set coefficients to 0 completely, the ridge penalty reduces the coefficient size more evenly. We will see that "why not both?" is also a legitimate suggestion.

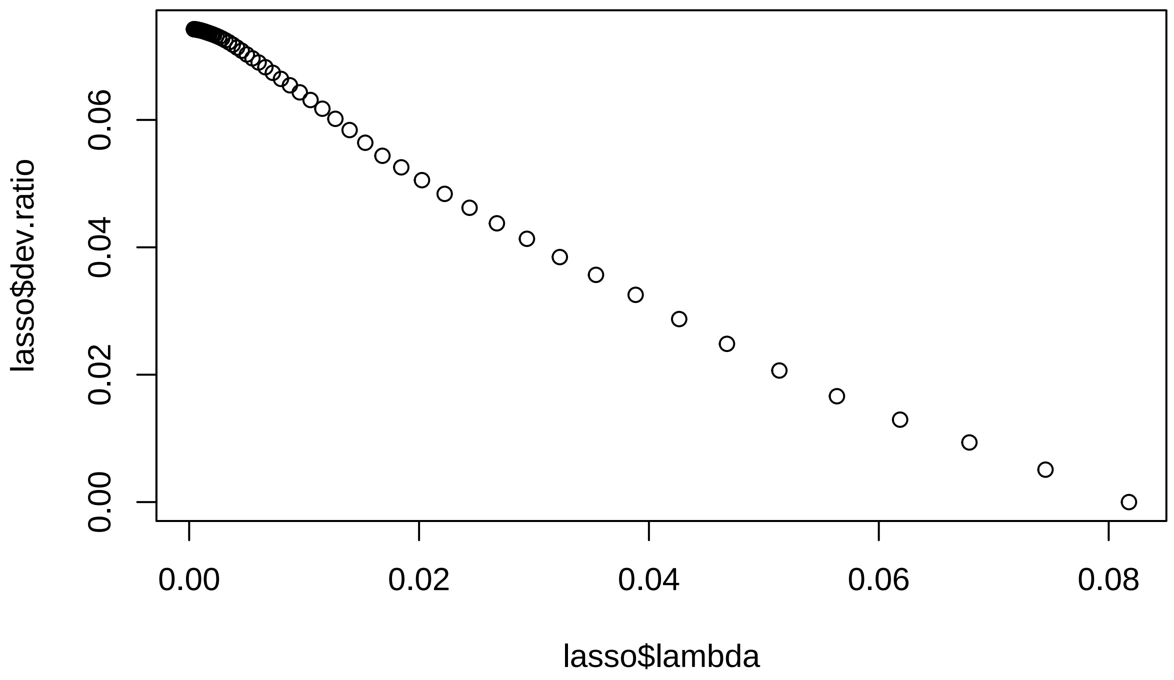
1. Load package *glmnet* and get familiar with it. Find out how to use it to fit a regularized logit model.
2. Fit a lasso-regularized logistic regression on the loans dataset. Make sure that the data is standardized when you fit the model.
3. Use the regularized model to make a prediction on the loans data. Note: If you have estimated the model for more than one value of the regularization weight lambda (the default), find a way to pick a reasonable value.

```
##
## Call:  glmnet(x = x, y = y, family = "binomial", alpha = 1, nlambda = 100,      standardize = TRUE)
##
##       Df      %Dev    Lambda
##  [1,]  0 8.083e-15 0.0817600
##  [2,]  1 5.090e-03 0.0745000
##  [3,]  1 9.359e-03 0.0678800
##  [4,]  1 1.294e-02 0.0618500
##  [5,]  2 1.662e-02 0.0563600
##  [6,]  3 2.066e-02 0.0513500
##  [7,]  3 2.484e-02 0.0467900
##  [8,]  4 2.873e-02 0.0426300
##  [9,]  4 3.254e-02 0.0388400
## [10,]  4 3.568e-02 0.0353900
## [11,]  5 3.847e-02 0.0322500
## [12,]  5 4.134e-02 0.0293800
## [13,]  5 4.377e-02 0.0267700
## [14,]  6 4.620e-02 0.0244000
## [15,]  7 4.839e-02 0.0222300
## [16,]  7 5.054e-02 0.0202500
## [17,]  8 5.256e-02 0.0184500
## [18,]  9 5.437e-02 0.0168100
## [19,] 10 5.642e-02 0.0153200
## [20,] 10 5.841e-02 0.0139600
## [21,] 12 6.017e-02 0.0127200
## [22,] 12 6.176e-02 0.0115900
## [23,] 13 6.313e-02 0.0105600
## [24,] 14 6.433e-02 0.0096220
## [25,] 15 6.544e-02 0.0087670
## [26,] 16 6.644e-02 0.0079880
## [27,] 18 6.739e-02 0.0072790
## [28,] 19 6.827e-02 0.0066320
## [29,] 19 6.902e-02 0.0060430
## [30,] 21 6.967e-02 0.0055060
## [31,] 22 7.027e-02 0.0050170
## [32,] 22 7.086e-02 0.0045710
## [33,] 23 7.136e-02 0.0041650
## [34,] 24 7.181e-02 0.0037950
## [35,] 24 7.219e-02 0.0034580
## [36,] 24 7.251e-02 0.0031510
## [37,] 24 7.277e-02 0.0028710
## [38,] 24 7.300e-02 0.0026160
## [39,] 24 7.318e-02 0.0023830
## [40,] 25 7.334e-02 0.0021720
## [41,] 25 7.348e-02 0.0019790
## [42,] 25 7.359e-02 0.0018030
## [43,] 25 7.368e-02 0.0016430
## [44,] 26 7.377e-02 0.0014970
## [45,] 26 7.386e-02 0.0013640
## [46,] 26 7.393e-02 0.0012430
## [47,] 26 7.398e-02 0.0011320
## [48,] 27 7.403e-02 0.0010320
## [49,] 27 7.408e-02 0.0009401
## [50,] 27 7.411e-02 0.0008566
## [51,] 27 7.414e-02 0.0007805
## [52,] 27 7.417e-02 0.0007111
## [53,] 27 7.419e-02 0.0006480
## [54,] 27 7.420e-02 0.0005904
## [55,] 27 7.422e-02 0.0005379
## [56,] 27 7.423e-02 0.0004902
## [57,] 27 7.424e-02 0.0004466
## [58,] 27 7.425e-02 0.0004069
```

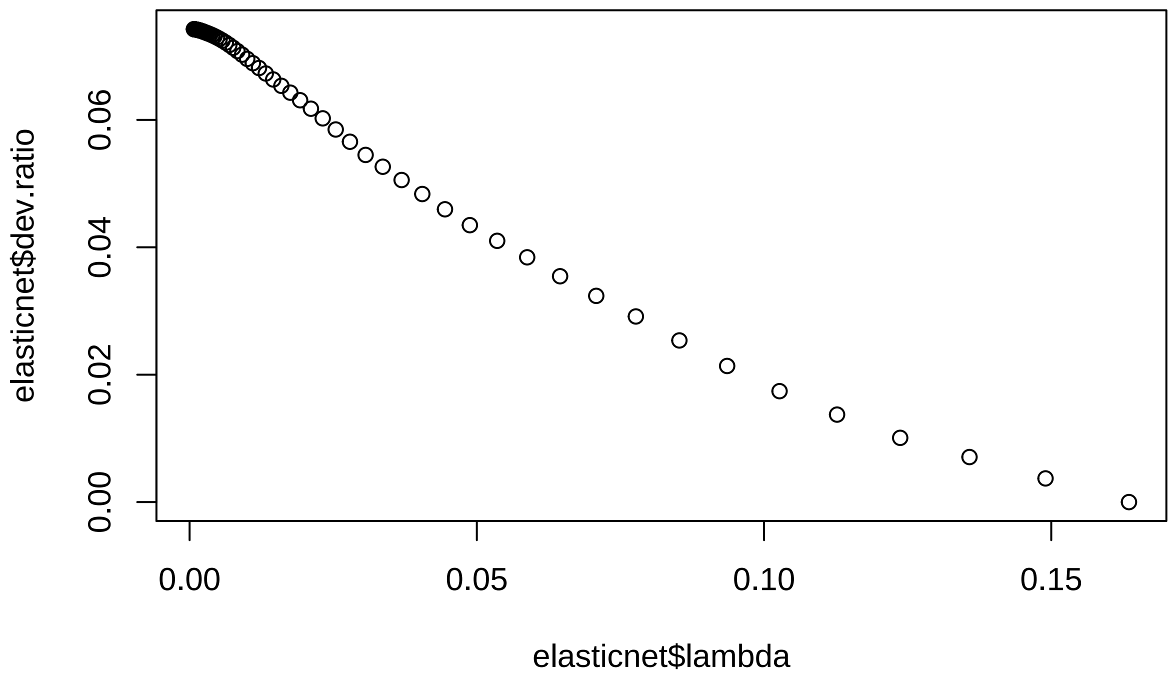




```
## List of 13
##  $ a0        : Named num [1:58] -1.027 -0.977 -0.932 -0.89 -0.858 ...
##   ..- attr(*, "names")= chr [1:58] "s0" "s1" "s2" "s3" ...
##  $ beta      :Formal class 'dgCMatrix' [package "Matrix"] with 6 slots
##   .. ..@ i       : int [1:966] 16 16 16 10 16 0 10 16 0 10 ...
##   .. ..@ p       : int [1:59] 0 0 1 2 3 5 8 11 15 19 ...
##   .. ..@ Dim     : int [1:2] 28 58
##   .. ..@ Dimnames:List of 2
##   .. .. ..$ : chr [1:28] "YOB" "nKIDS" "nDEP" "PHON" ...
##   .. .. ..$ : chr [1:58] "s0" "s1" "s2" "s3" ...
##   .. ..@ x       : num [1:966] -2.37e-06 -4.54e-06 -6.55e-06 3.09e-02 -8.28e-06 ...
##   .. ..@ factors : list()
##  $ df        : int [1:58] 0 1 1 1 2 3 3 4 4 4 ...
##  $ dim       : int [1:2] 28 58
##  $ lambda    : num [1:58] 0.0818 0.0745 0.0679 0.0619 0.0564 ...
##  $ dev.ratio : num [1:58] 8.08e-15 5.09e-03 9.36e-03 1.29e-02 1.66e-02 ...
##  $ nulldev   : num 1413
##  $ npasses   : int 341
##  $ jerr      : int 0
##  $ offset    : logi FALSE
##  $ classnames: chr [1:2] "good" "bad"
##  $ call      : language glmnet(x = x, y = y, family = "binomial", alpha = 1, nlambda = 100,      star
##  $ nobs      : int 1225
##  - attr(*, "class")= chr [1:2] "lognet" "glmnet"
```



```
## 29 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept)  5.104080e-02
## YOB         -1.278641e-02
## nKIDS        .
## nDEP         .
## PHON        -1.335886e-03
## dINC_SP     -2.289582e-05
## EMPS_AB      .
## EMPS_AE      .
## EMPS_AM      .
## EMPS_AN      .
## EMPS_AP     -1.062751e-02
## EMPS_AR      7.542929e-02
## EMPS_AT      .
## EMPS_AU      1.461959e-01
## EMPS_AV     -1.137986e-01
## EMPS_AW      1.888779e-01
## EMPS_AZ      .
## dINC_A      -2.232406e-05
## RESN         6.260033e-01
## RESO         1.370932e-01
## RESP         .
## RESU         .
## dHVAL        .
## dMBO         .
## dOUTM        .
## dOUTL        1.381870e-05
## dOUTHP      -6.311260e-04
## dOUTCC      -1.027154e-03
## YOB_missing  .
```



```
## 29 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept)  1.193657e-01
## YOB         -1.304775e-02
## nKIDS        .
## nDEP         1.137832e-01
## PHON        -6.646943e-02
## dINC_SP     -3.177238e-05
## EMPS_AB      9.241685e-02
## EMPS_AE      2.337465e-02
## EMPS_AM      .
## EMPS_AN     -4.291716e-02
## EMPS_AP     -5.153650e-02
## EMPS_AR      1.090945e-01
## EMPS_AT      .
## EMPS_AU      3.786922e-01
## EMPS_AV     -1.826360e-01
## EMPS_AW      3.231404e-01
## EMPS_AZ      .
## dINC_A      -2.311086e-05
## RESN         7.210407e-01
## RESO         1.930485e-01
## RESP         .
## RESU        -3.127124e-02
## dHVAL       -5.432122e-08
## dMBO         9.638815e-07
## dOUTM        4.871229e-05
## dOUTL        3.517760e-05
## dOUTHP      -9.190368e-04
## dOUTCC      -1.405838e-03
## YOB_missing  .
```

```
##          lr lasso elasticnet
##  [1,]  0.59  0.05       0.12
##  [2,] -0.02 -0.01      -0.01
##  [3,] -0.04  0.00       0.00
##  [4,]  0.26  0.00       0.11
##  [5,] -0.14  0.00      -0.07
##  [6,]  0.00  0.00       0.00
##  [7,] -0.20  0.00       0.09
##  [8,] -0.23  0.00       0.02
##  [9,] -0.77  0.00       0.00
## [10,] -0.36  0.00      -0.04
## [11,] -0.24 -0.01      -0.05
## [12,] -0.34  0.08       0.11
## [13,]  0.31  0.00       0.00
## [14,] -0.53  0.15       0.38
## [15,]  0.19 -0.11      -0.18
## [16,] -0.48  0.19       0.32
## [17,]  0.00  0.00       0.00
## [18,]  0.94  0.00       0.00
## [19,]  0.44  0.63       0.72
## [20,]  0.11  0.14       0.19
## [21,] -0.03  0.00       0.00
## [22,]  0.00  0.00      -0.03
## [23,]  0.00  0.00       0.00
## [24,]  0.00  0.00       0.00
## [25,]  0.00  0.00       0.00
## [26,]  0.00  0.00       0.00
## [27,]  0.00  0.00       0.00
## [28,]  0.30  0.00       0.00
## [29,]  0.59  0.00       0.00
```