Analytical Standards for Regression-based Predictive Analytics : Methodologies, Naming Conventions and Coding Practices

Daniel Chertok, Ari Robicsek
Clinical Analytics Team
©NorthShore University HealthSystem

6th February, 2020

Abstract

This document proposes a set of methodology and programming standards for the Clinical Analytics team. It is intended as a set of guidelines that will be developed over time as the needs of the team evolve. Guidelines include a review of statistics, survival analysis, general programming techniques, naming conventions, R coding practices and a general approach to tackling most common types of applied predictive analysis handled by the team.

Contents

List of Figures

List of Tables

Listings

1 Introduction

Predictive modeling tasks handled by Clinical Analytics fall into one of two categories: *classification* and *regression* . Classification answers the question, "What group of patients does this individual belong to?" Its outcome is a categorical - quite often, binary - variable. Regression answers the question, "How much or how many?" Its outcome is a cardinal variable. While the outcomes of these two types of mathematical models are different, the underlying methodologies are very similar and are considered in Sections 3.3 and 3.2. A combination approach may be appropriate for problems requiring the development of quantification metrics for events of interest: first, identify (or classify) potential outcomes, then evaluate the impact of each outcome separately. In this case, a classification algorithm should be followed with a regression; more often than not, quantification of only the positive outcome is of interest to us.

An important - though sometimes overlooked - step in streamlining the research and development (R&D) methodology is agreeing on standardized terminology for the research process. A well-developed glossary of terms (see Section 2) can assure that identical tasks or processes are described in identical terms, a concept similar to "data integrity" as defined by the principles of database design [6].

Anecdotal evidence suggests that a data scientist (whatever this term currently entails) spends 90% of her time scrubbing the data [29] and only 10% of it doing what she learned in her school's Advanced Scientific Fortunetelling program. Like an experienced cook who appreciates the role of quality ingredients in meal preparation, a sensible data scientist may be able to achieve good results by simply ensuring that the data ingested into her algorithm is clean. Agreed-upon procedures (AUP) for data cleansing and storage are covered in 3.5 and 5.1.

Consistent, scalable development of reliable and reusable software is an important part of introducing the developed methodologies into production. A collection of good coding practices relevant to predictive analytic development is presented in Section 6. A good foundation for developing robust code and assuring business continuity includes

- proper revision control practices (6.1),
- accessible and consistently named code repositories and development sandboxes (6.2) and
- readable and transparent code modules (6.4, 6.5) in R (6.5.1).

Once an algorithm has been prototyped and implemented to the developers' satisfaction, the responsibility for putting it into everyday use it shifts to the production team. The process of testing, validation and verification can be drawn out and contentious unless the rules of the game are well defined in advance. Efficient practices for lightening the burden on both the original developers and the QA team are described in Section 6.6.

Finally, once the results have been validated, consistent and easy to understand presentation can facilitate their acceptance by the intended audience. Appropriate standards are covered in Section 4.

Examples contained in this manual are based on real data, however, in order to protect potentially sensitive information, the numbers have been modified and the names of the entities involved obscured where deemed necessary.

2 Preferred terminology

Nomenclature of terms

indicator variable

a variable with only two outcomes: 0 and 1, FALSE and TRUE 29, 106

binary variable

see indicator variable

dummy variable

see indicator variable

categorical variable

a variable with two or more outcomes whose values cannot be meaningfully converted into comparable numbers, e.g., ethnicity, gender, geographical region *see also* nominal variable & ordinal variable, 9, 29, 106, 107, 108

ordinal variable

a categorical variable whose values can be meaningfully ordered but not quantitatively compared, e.g., stage of cancer, education level, degree of satisfaction 29, 106, 107, 108

interval variable

a discrete variable whose values are equidistant and the zero is arbitrarily set, e.g., IQ scores, number of hospital admissions, number of children in a family *see also* continuous variable, 29, 106, 107, 108

continuous variable

a variable whose values can take on any (real) number, e.g., body mass index, systolic blood pressure, hemoglobin 29

Type I error

an incorrect rejection of the null hypothesis *see also* false positive

t-test

Student's t-test 106, 107

true positive

an event correctly classified as having an outcome of interest 10, 11

false positive

an event incorrectly classified as having an outcome of interest *see also* Type I error, 10, 11, 79

true negative

an event correctly classified as not having an outcome of interest 11

false negative

an event incorrectly classified as not having an outcome of interest *see also* Type II error, 10, 11, 79

positive predictive value

also precision
$\text{PPV}=\frac{\text{number of true positives}}{\text{number of predicted positives}}$
$=\frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false positives}}$ 11, 80, 82

relative survival probability

survival probability of a group under consideration relative to that of a benchmark group, e.g., survival probability of cancer patients relative to the general population of the same age 27

survival analysis

a branch of statistics (or, more broadly, applied mathematics) concerned with predicting failure events among a given population or group of technical objects 25

time since first diagnosis

interval of time between the first recorded or implied diagnosis assigned to the patient and the time t of interest (e.g., current time) 27

true positive rate

also sensitivity, hit rate, recall
$\text{TPR}=\frac{\text{number of true positives}}{\text{number of true positives} + \text{number of false negatives}} = \frac{\text{number of true positives}}{\text{total positive outcomes}}$
11, 73, 78, 79, 80

specificity

also true negative rate
TNR=$\frac{\text{true negative}}{\text{false positive}+\text{true negative}} = \frac{\text{true negative}}{\text{total negative outcomes}}$ 73, 78, 79, 82, 84

false positive rate

also fallout
FPR=$\frac{\text{false positive}}{\text{false positive}+\text{true negative}} = \frac{\text{false positive}}{\text{total negative outcomes}}$ 11

receiver operating characteristic (ROC) curve

A curve that visualizes the accuracy of a classification algorithm as a relationship between true positive rate and false positive rate 4, 11, 77, 80

lift curve

A curve that visualizes the relationship between true positive rate and the fraction of the population targeted by the response solicitation campaign. It is a variation on the receiver operating characteristic (ROC) curve 4, 80, 81

lift

Lift $=\frac{\text{\% of outcomes of interest in the population selected by the model}}{\text{\% of outcomes of interest in the whole population}}$ 81, 82

F_1 score

$F_1 = 2\frac{\text{positive predictive value}\times\text{true positive rate}}{\text{positive predictive value}+\text{true positive rate}}$ 4, 29, 69, 71, 73, 74, 78, 80, 84

Matthews' correlation coefficient

$MCC = \frac{TP\times TN-FP\times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}}$, where TP is the number of true positives, TN is the number of true negatives, FP is the number of false positives and FN is the number of false negatives 4, 29, 69, 71, 73, 74, 78, 80, 84

interaction term

also cross term
in a (generalized) linear model, a nonlinear term of the form $\prod_{i=1}^{m} X_i$, where X_i is the i-th predictive variable, m the order of nonlinearity; the simplest nontrivial ($m = 2$) case being $X_1 \times X_2$ 68, 69

3 Methodology

3.1 General approach

An outline of a general approach to solving an analytical problem is presented in Fig. 3.1.

As mentioned in Section 1, the majority of predictive analytic problems can be solved by employing one of two wide types of forecasting methodologies: regression and classification. Regression[1] should be used when the output variable[2] is interval or continuous, i.e., can take on any permissible value inside an interval (which may include the whole real axis). Examples of this type of problem include predicting:

1. a lab test result based on the patient's demographics, clinical history and other lab tests;

2. the number of admissions based on the previous history and calendar data;

3. patient management cost based on patient's data.

Logistic regression is one of the most widely practically used classification algorithms. It is easy to implement[3], intuitive and can be made sufficiently accurate for most uncomplicated modeling tasks. Mathematically similar to linear regression, it[4] can (and often should) be used when the output variable is an indicator, binary, categorical, nominal or ordinal. Examples of this type of problem include

1. predicting patient's risk of mortality, admission or readmission based on demographic and clinical data;

2. classifying the severity of a patient's condition based on available clinical data and history;

3. identifying those patients among high risk population who are most likely to respond to intervention [3].

[1] Or another appropriate fitting technique

[2] Scalar, i.e., single-valued or vector, i.e., multivalued.

[3] In fact, it is commonly implemented in many statistical packages and programming languages, including `R`.

[4] Or another appropriate classification technique.

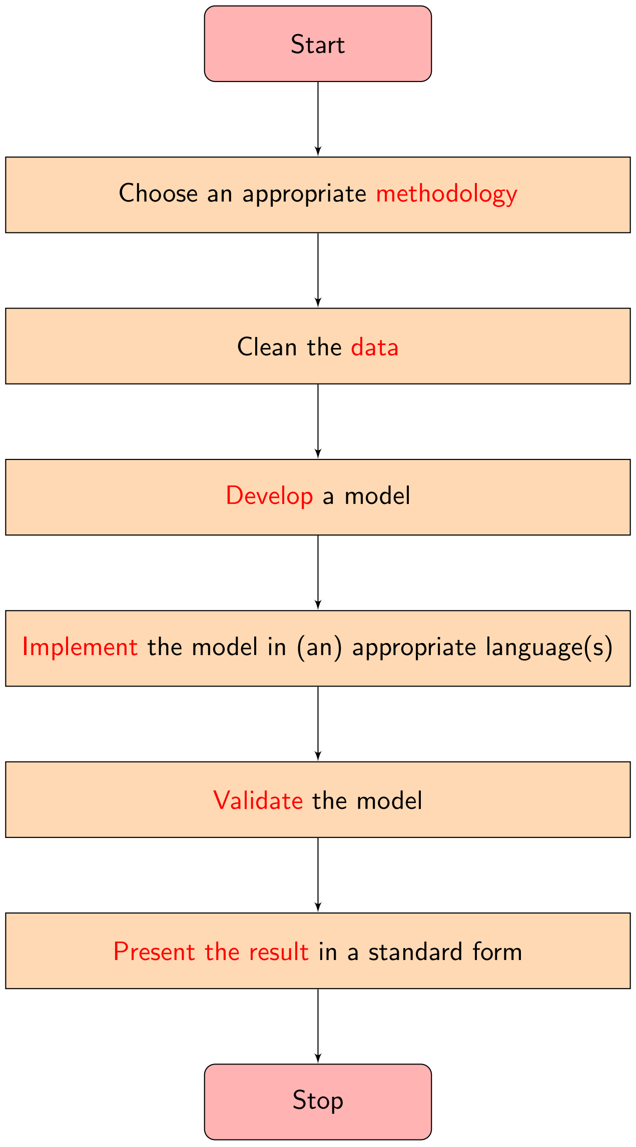

Start
Choose an appropriate methodology
Clean the data
Develop a model
Implement the model in (an) appropriate language(s)
Validate the model
Present the result in a standard form
Stop

Figure 3.1: A generic algorithm for proceeding with an analytical project within the framework of Clinical Analytics.

3.2 Regression

A regression problem answers the question "How much output quantity or how many items or events can be generated as a result of the process under investigation?". The solution of a regression problem can be found as a result of an optimization algorithm on a measure of the difference between predicted and actual outputs. The simplest model in this case, *linear regression*, assumes that

- there is no (measurement) error in the values of predictors and the dependent variable;
- predictor variables are
  - (statistically) independent;
  - linearly independent (not collinear), i.e., the matrix of predictors has full rank (this is separate from the condition above);
- a linear relationship of the type

  $$Y_i = \beta_0 + \sum_{i=1}^{M} X_{ij}\beta_j \ , i = \overline{1, N} \ , \tag{3.1}$$

  where N is the number of observations and M is the number of predictive variables, exists between the predictors and the output variable;
- residuals, or errors, i.e., differences between observed and predicted values of the dependent variable

  $$\epsilon_i = Y_i - \hat{Y}_i \ , \tag{3.2}$$

  where $\hat{Y}_i, \ i = \overline{1, N}$ are the predicted and $Y_i, \ i = \overline{1, N}$ are the actual values of the dependent variable, are
  - distributed with a zero mean (exogeneity);
  - homoscedastic (of constant variance);
  - of finite variance;
  - statistically independent of one another and of the independent variables.

It is sometimes assumed that residuals are normally distributed, i.e.$\epsilon \sim \mathcal{N}\left(0, \sigma^2\right)$, however, this assumption may be relaxed through the application of the Central Limit Theorem if the number of observations is sufficiently large (exceeds the proverbial $N = 30$). In this simplest case, an analytical solution exists and can be found using readily available formulas. A more complex problem can be reduced to linear regression if the functional form of relationship between the predictors and the output is known *apriori*, e.g., by taking the logarithm of the outcome variable, both sides of the equation or a combination of both (*loglinear regression*, *log-log regression* or *log-linear-log regression*).

Let us assume that a linear relationship between the predictors and the dependent variable described by (3.1) does exist. In the most general case, X is an $N \times M$ matrix of predictive variables, β is an $M \times 1$ vector and β_0 is a scalar:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1M} \\ \vdots & \ddots & \vdots \\ x_{N1} & \cdots & x_{NM} \end{bmatrix}, \tag{3.3}$$

$$\beta = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_M \end{bmatrix}, \tag{3.4}$$

$$Y = X\beta + \beta_0 \tag{3.5}$$

Instead of (3.3 - 3.5), we can consider an augmented $N \times (M+1)$ matrix X_* and $(M+1) \times 1$ vector β_* that together form an equivalent system of equations:

$$X = \begin{bmatrix} 1 & x_{11} & \cdots & x_{1M} \\ \vdots & \ddots & \vdots & \vdots \\ 1 & x_{N1} & \cdots & x_{NM} \end{bmatrix}, \tag{3.6}$$

$$\beta_* = \begin{bmatrix} \beta_0 \\ \beta_1 \\ \vdots \\ \beta_M \end{bmatrix}, \tag{3.7}$$

$$Y = X_*\beta_* . \tag{3.8}$$

Observe that the left hand side of (3.5) is exactly the same as the left hand side of (3.8) once we set $a_0 \equiv b$. For the ease of exposition we shall drop subindex $_*$ from X_* and β_* and continue to refer to these augmented variables as X and β, i.e.,

$$Y = X\beta . \tag{3.9}$$

The solution of (3.9) can be found in the form of

$$\beta = (X^T X)^{-1} X^T y \ , \tag{3.10}$$

where X^T is the transposed X ($X^T_{ij} = X_{ji}$), provided that $(X^T X)^{-1}$ exists (see, e.g., [21])

An example of a fitted curve is presented in Fig. 3.2

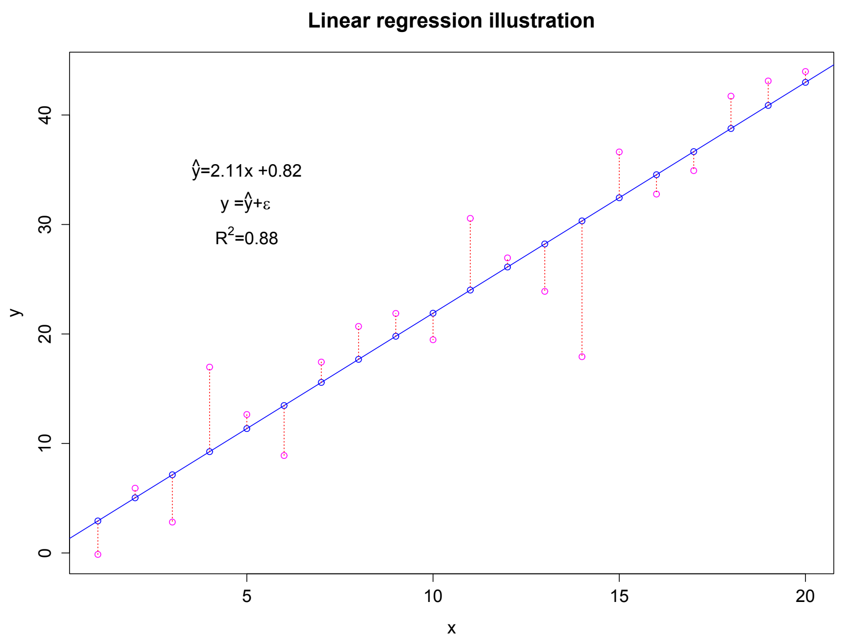


Figure 3.2: Linear regression: an illustration

A common metric for assessing the quality (or *goodness-of-fit*) of linear regression is its *coefficient of determination* R^2, defined as

$$R^2 = 1 - \frac{Var(\epsilon)}{Var(y)} = 1 - \frac{\frac{1}{N}\sum_{i=1}^{N} \epsilon_i^2}{\frac{1}{N}\sum_{i=1}^{N}(y_i - \overline{y})^2} \ , \tag{3.11}$$

since we assume $E(\epsilon) = 0$. The coefficient of determination quantifies what fraction (percentage) of the variation of the dependent variable, y, can be explained by the variation of the independent variable(s), x (via x's linear relationship to y).

A general approach to attacking regression problems is presented in Fig. 3.3. In `R` an ordinary linear regression model can be built using function `lm`.

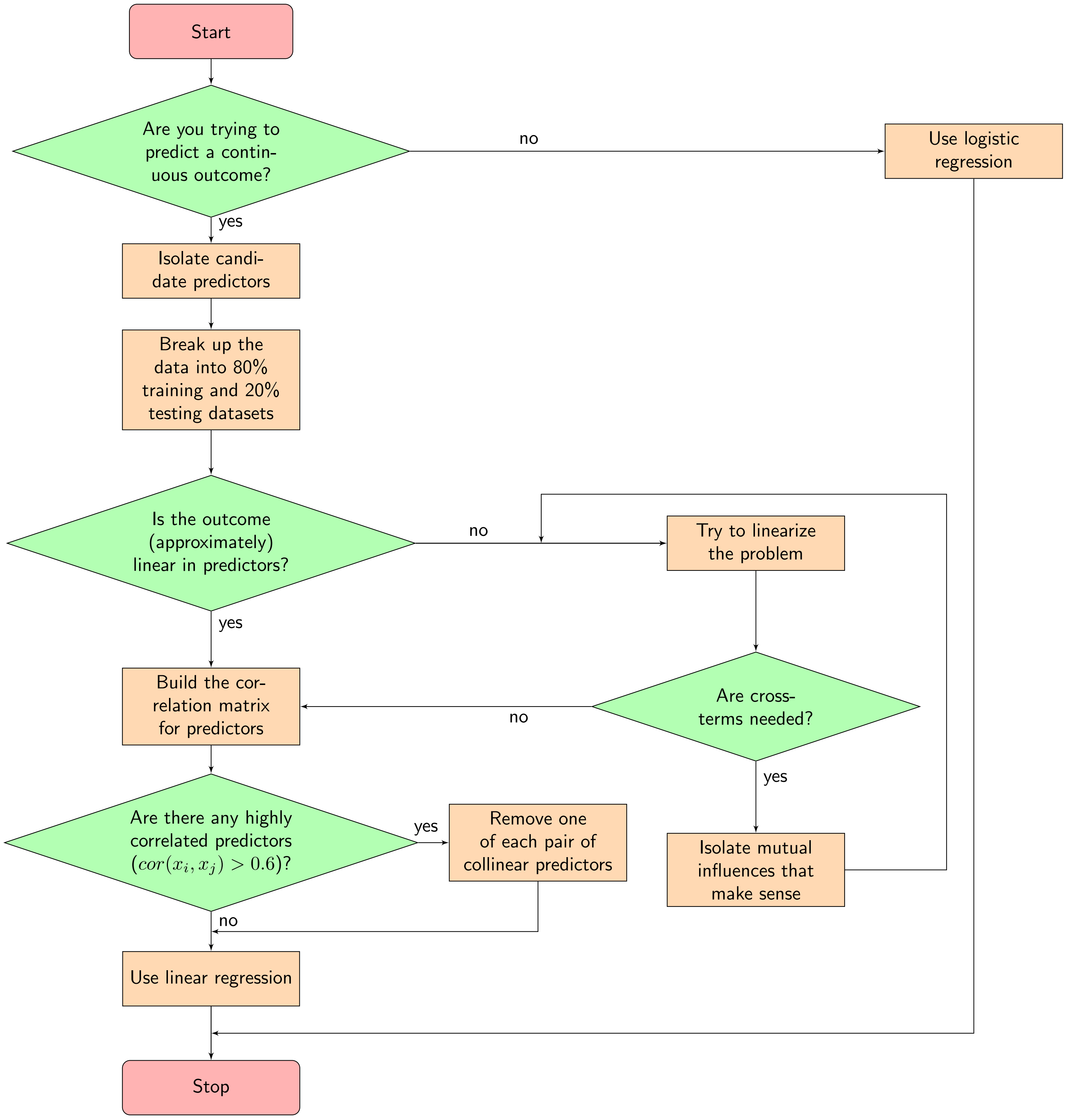

Start
Are you trying to predict a continuous outcome?
no
Use logistic regression
yes
Isolate candidate predictors
Break up the data into 80% training and 20% testing datasets
Is the outcome (approximately) linear in predictors?
no
Try to linearize the problem
yes
Are cross-terms needed?
no
Build the correlation matrix for predictors
yes
Isolate mutual influences that make sense
Are there any highly correlated predictors ($cor(x_i, x_j) > 0.6$)?
yes
Remove one of each pair of collinear predictors
no
Use linear regression
Stop

Figure 3.3: Decision tree for linear regression problems.

As an example, consider the data presented in Table 3.1.

Predictive variable	Outcome	Predictive variable	Outcome	Predictive variable	Outcome
1	-0.1322691	8	20.6916235	15	36.6246546
2	5.9182166	9	21.8789068	16	32.7753320
3	2.8218569	10	19.4730581	17	34.9190487
4	16.9764040	11	30.5589058	18	41.7191811
5	12.6475389	12	26.9492162	19	43.1061060
6	8.8976581	13	23.8937971	20	43.9695066
7	17.4371453	14	17.9265006		

Table 3.1: Example: Logistic regression

The corresponding R code is presented in Listing 3.1.

```
set.seed(1)
x <- 1:20
xR <- rnorm( 1:20 )
y <- 2 * x + 1 + 5 * xR
lm <- lm( y ~ x )
coef <- coef( lm )
yHat <- x * coef[2] + coef[1]
plot( x, y, main="Linear regression illustration", col
    ='magenta' )
abline( coef=coef, col='blue' )
points( x, yHat, type='p', col='blue' )
for( i in 1:length( x ) ) {
  segments( x[i], yHat[i], x[i], y[i], lty='dotted',
      col='red', pch=16 )
}
a <- sprintf( "%.2f", coef[2] )
b <- sprintf( "%.2f", coef[1] )
text( 5, 35, bquote( paste( hat( y ), "=", .(a), "x +"
    , .(b) ) ) )
text( 5, 32, bquote( paste( "y =", hat( y ), "+",
    epsilon ) ) )
R2 <- sprintf( "%.2f", summary(lm)$r.squared )
text( 5, 29, bquote( paste( R^2, "=", .(R2) ) ) )
```

```
summary(lm)
confint(lm)
```

Listing 3.1: Example: R code for linear regression.

producing the output in Listing 3.2.

```
Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-12.4038  -2.5841   0.9373   2.4165   7.7252

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   0.8195     2.1579    0.38    0.709
x             2.1079     0.1801   11.70 7.56e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
    0.1 ' ' 1

Residual standard error: 4.645 on 18 degrees of
    freedom
Multiple R-squared:  0.8838,    Adjusted R-squared:
    0.8774
F-statistic: 136.9 on 1 and 18 DF,  p-value: 7.56e-10

                 2.5 %   97.5 %
(Intercept) -3.714035 5.353106
x            1.729458 2.486368
```

Listing 3.2: Example: R linear regression model summary output.

It follows from Listing (3.2) that only the slope estimate is statistically significantly different from 0 at the 95% level ($p < 0.05$). The 95% confidence interval for the intercept is $[-3.71; 5.35]$, and for the slope it is $[1.73; 2.49]$. The coefficient of determination $R^2 = 0.88$ indicates a very good fit between the data points and the developed linear model which has the form[1]

$$\hat{y} = 0.8195 + 2.1079x \, , \tag{3.12}$$

[1]The accuracy of coefficient estimates is lower than 0.5×10^{-4}

or, if we accept the null hypothesis concerning the intercept ($\beta_0 = 0$),

$$\hat{y} \quad = \quad 2.1079x \; . \tag{3.13}$$

With the benefit of foresight[1], we already presented an illustration of (3.12) in Fig. 3.2.

3.3 Using logistic regression for classification problems

A classification problem answers the question "What group does this object belong to?". The answer can depend on the available data as described in Fig. 3.4. Quite often, the easiest-to-implement method is preferable since it can be deployed with the least effort and does not require infrastructure and process adjustments. For practical purposes, logistic regression and its modifications often

- represent a good trade-off between cost and accuracy,

- make the contribution of different explanatory variables easy to understand and

- yield results that easy to interpret,

and therefore should be implemented whenever possible.

[1]Authors of predictive analytics literature have frequently enjoyed this advantage [8].

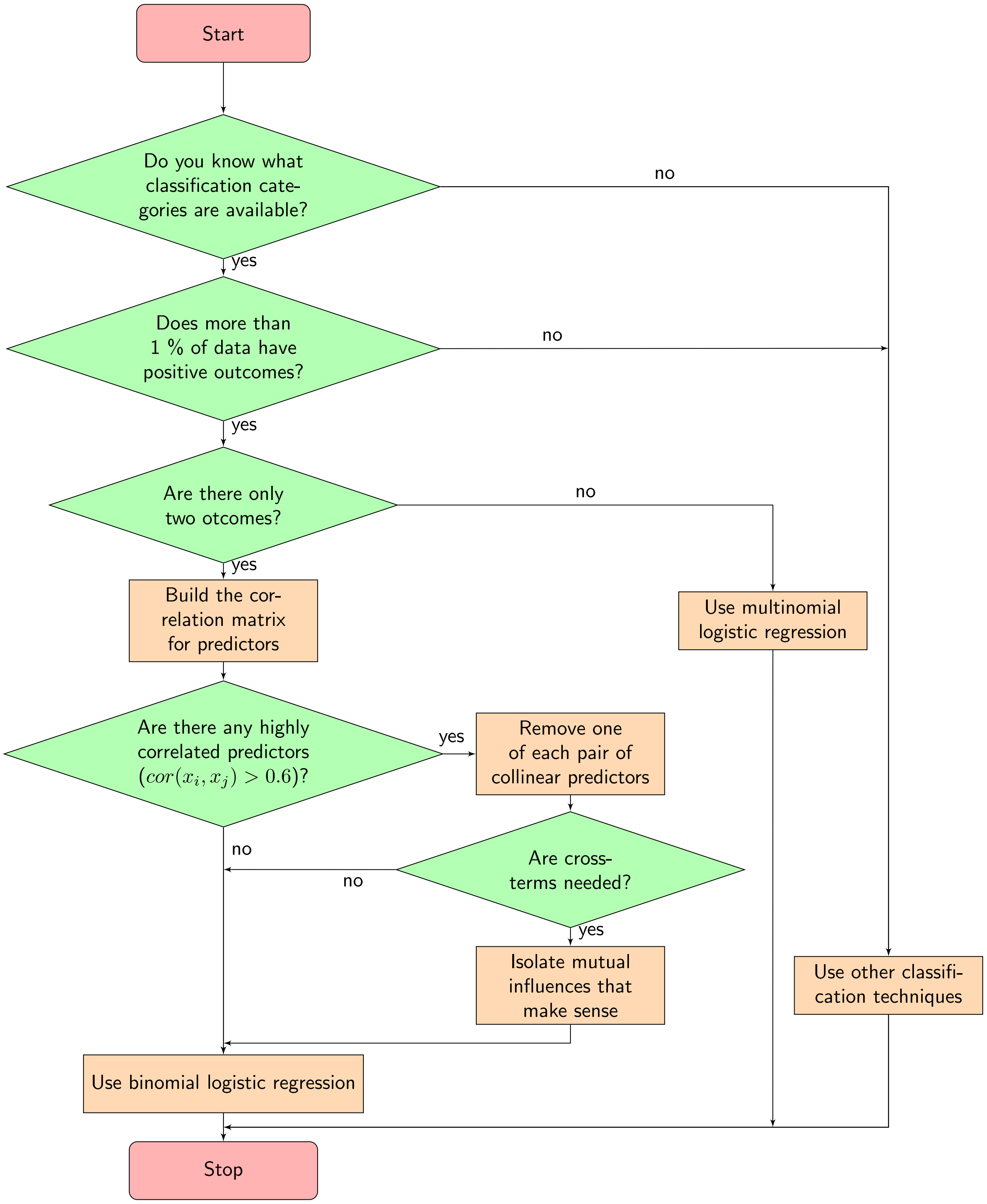

Start
Do you know what classification categories are available?
no
yes
Does more than 1 % of data have positive outcomes?
no
yes
Are there only two otcomes?
no
yes
Build the correlation matrix for predictors
Use multinomial logistic regression
Are there any highly correlated predictors ($cor(x_i, x_j) > 0.6$)?
yes
Remove one of each pair of collinear predictors
no
Are cross-terms needed?
no
yes
Isolate mutual influences that make sense
Use other classification techniques
Use binomial logistic regression
Stop

Figure 3.4: Decision tree for logistic regression classification problems.

The mathematical rationale behind logistic regression is based on mapping the probability domain onto the real axis as outlined below:

$$Y|X \sim \mathrm{B}(1,p) \;, \tag{3.14}$$

$$p(x) = P(Y|X) \tag{3.15}$$

$$logit(p(x)) = \ln \frac{p(x)}{1-p(x)} \;, p \in [0;1] \;. \tag{3.16}$$

$$\tag{3.17}$$

In applying transformation (3.16), one must effectively hold out hope of approximating a discrete-valued function with a smooth sigmoid function defined by (3.16) as illustrated in Fig 3.5.

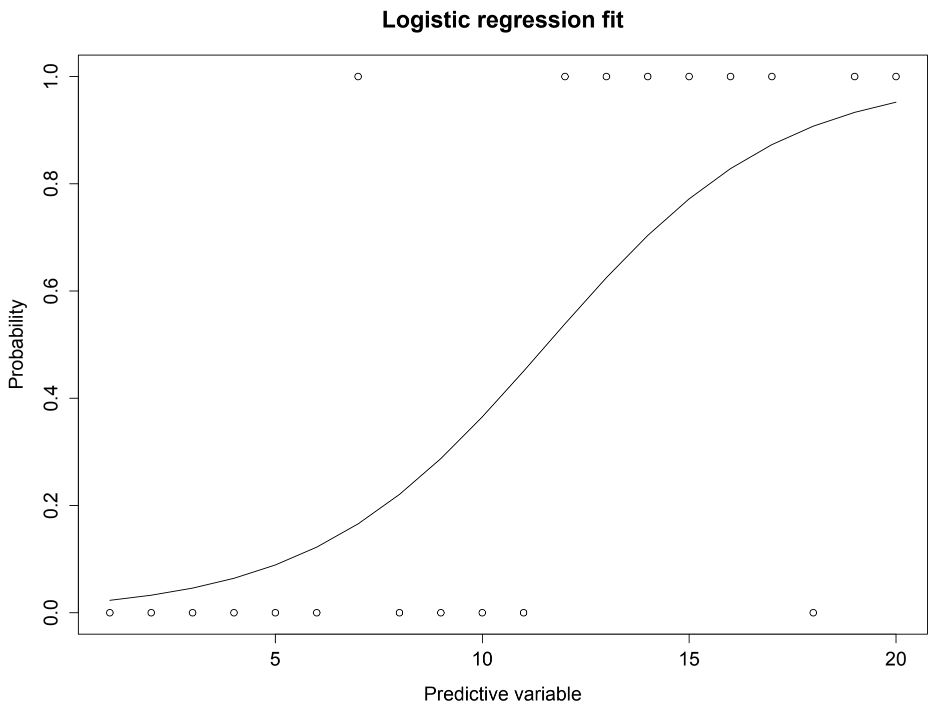


Figure 3.5: Logistic regression fit: an illustration

The accuracy of approximation (3.18) depends on the separability of two sets, $Y = 0$ and $Y = 1$.

Logistic regression model corresponding to (3.16) has the form

$$logit(p(x)) = X\beta \,, logit(p(x)) \in [-\infty, \infty] \;, \tag{3.18}$$

however, coefficients β cannot be found using linear regression techniques described in Section 3.2 since the observed outcome of interest ($p = 1$) corresponds to positive infinity in the transformed range of (3.18) and the remainder of cases ($p = 0$) correspond to negative infinity. The solution of (3.18) is found using the *maximum likelihood method* [15]. Observe from (3.16) that

$$p(x) = \frac{1}{1 + e^{-X\beta}} . \tag{3.19}$$

The likelihood of obtaining outcome y_i given the value of the predictor variable x_i is $p(x_i)$

$$P(y_i|x_i) = p(x_i)^{y_i} \left[1 - p(x_i)\right]^{1-y_i} , \tag{3.20}$$

and the total likelihood of obtaining a specific sequence of outcomes is

$$l(\beta) = \prod_{i=1}^{N} p(x_i)^{y_i} \left[1 - p(x_i)\right]^{1-y_i} , \tag{3.21}$$

or, taking the natural logarithm of both sides for convenience,

$$L(\beta) = \ln\left[l(\beta)\right] = \sum_{i=1}^{N} \left\{y_i \ln p(x_i) + (1 - y_i) \ln\left[1 - p(x_i)\right]\right\} , \tag{3.22}$$

The maximum of $L(\beta)$ can be found by differentiating (3.22) with respect to β_i and setting the resulting equations to 0:

$$\frac{\partial L(\beta)}{\partial \beta_i} = \sum_{i=1}^{N} x_i \left[y_i - p(x_i)\right] = 0 . \tag{3.23}$$

The solution of (3.23) can be found by standard numerical solution techniques, e.g., the Newton-Raphson method [28]. Fortunately, Open Source and commercial statistical software has logistic regression algorithms efficiently implemented, so that implementing the above-mentioned algorithm is not a concern for a typical user.

As an example, consider a classification problem described by Table 3.2.

Predictive variable	Outcome	Predictive variable	Outcome	Predictive variable	Outcome
1	0	8	0	15	1
2	0	9	0	16	1
3	0	10	0	17	1
4	0	11	0	18	0
5	0	12	1	19	1
6	0	13	1	20	1
7	1	14	1		

Table 3.2: Example: Logistic regression

The corresponding R code is presented in Listing 3.3.

```
x <- 1:20
y <- c(rep(0, 6), 1, rep(0, 4), rep(1, 6), 0, 1, 1)
glm <- glm(y ~ x, family=binomial( link='logit'))
summary(glm)
confint(glm)
```

Listing 3.3: Example: R code for logistic regression.

producing the output in Listing 3.4.

```
Call:
glm(formula = y ~ x, family = binomial(link = "logit")
    )

Deviance Residuals:
    Min         1Q     Median        3Q       Max
-2.1815    -0.5596    -0.2371    0.6403    1.8960

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -4.0972     1.7830  -2.298   0.0216 *
x             0.3544     0.1476   2.401   0.0163 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.'
    0.1 ' ' 1
```

```
(Dispersion parameter for binomial family taken to be
    1)

    Null deviance: 27.526  on 19  degrees of freedom
Residual deviance: 16.707  on 18  degrees of freedom
AIC: 20.707

Number of Fisher Scoring iterations: 5

Waiting for profiling to be done...
                  2.5 %      97.5 %
(Intercept) -8.6739420  -1.2930813
x            0.1207693   0.7329936
```

Listing 3.4: Example: R logistic regression model summary output.

It follows from Listing (3.4) that both the intercept and slope estimates are statistically significantly different from 0 at the 95% level ($p < 0.05$). The 95% confident interval for the intercept is $[-8.67; -1.29]$, and for the slope it is $[0.12; 0.73]$. The resulting model has the form[1]

$$logit(\hat{p}(x)) \quad = \quad -4.0972 + 0.3544x \ , \tag{3.24}$$

and the probability estimate is

$$\hat{p}(x) \quad = \quad \frac{1}{1 + e^{4.0972 - 0.3544x}} \ . \tag{3.25}$$

As in Section 3.2, we already presented an illustration of (3.25) in Fig. 3.5.

3.4 Survival modeling

A frequently asked question in healthcare analytics is: "What is the probability of survival for (at least) time t from now ($t_0 = 0$)of an individual with specific conditions?" or, conversely, "What is the expected survival time of a given individual?". Survival analysis is a form of regression that can help answer these questions. A standard procedure for evaluating survival probability, and, to some extent, expected survival time, is *Cox survival analysis* [12], [18], [27]. At the core of it is the semiparametric *Cox proportional hazard model.*

[1]As in Section 3.2, the accuracy of coefficient estimates is lower than 0.5×10^{-4}

In the following analysis, we assume that an outcome of interest represents an irreversible state transition (e.g., alive to dead). The probability of an event of interest occurring before time t is

$$P(t) \quad = \quad Pr(T \leq t) = \int_0^t p(x)dx \ , \tag{3.26}$$

where T is the time of the event and $p(x)$ is the probability density of the (possibly unknown) distribution of such an event. The probability of an individual surviving until (at least) time$t$ is termed the *survival function* and represents the complement of $P(t)$

$$S(t) \quad = \quad Pr(T > t) = \int_t^\infty p(x)dx \ . \tag{3.27}$$

The rate of arrival of outcomes of at time t is equal to the instantaneous probability of an event at time t conditional upon surviving until that time and can be calculated as[1]

$$\begin{aligned} h(t) &= \lim_{\Delta t \to 0} \frac{P(t \leq T < t + \Delta t | T \geq t)}{\Delta t} \\ &= \frac{dP(t)}{dt} \frac{1}{S(t)} = \frac{p(t)}{S(t)} \ . \end{aligned} \tag{3.28}$$

In the Cox model, hazard rate $h(t)$ is regressed against a set of predictors X_i as

$$h(t) \quad = \quad h_0(t) e^{\sum_{i=1}^N b_i x_i} \ , \tag{3.29}$$

where b_i is the weighting of x_i, the i-th of N explanatory variables. For the population of M individuals, (3.29) can be rewritten as

$$h_i(t) \quad = \quad h_{i_0}(t) e^{\sum_{j=1}^N b_j x_{ij}} \ , i = \overline{1:N}, j = \overline{1:M} \ . \tag{3.30}$$

Taking the (natural) logarithm of both sides of (3.30), we arrive at the equivalent of (3.18):

$$\ln \frac{h_i(t)}{h_{i_0}(t)} \quad = \quad \sum_{j=1}^N b_j x_{ij} \ , i = \overline{1:N}, j = \overline{1:M} \ . \tag{3.31}$$

The form of $h_{i_0}(t)$ is not formally specified; its shape is determined by empirical data in the training dataset giving rise to the unparametric portion

[1]conditional probability

of the model.[1] The solution of (3.26) is delivered by the maximum of the *partial likelihood function* defined in [5] as

$$
\begin{aligned}
L_p &= \prod_{i=1}^{N} \left[\frac{e^{x_i \beta}}{\sum_{j=1} N Y_{ij} e^{x_i \beta}} \right]^{\delta_i} , && (3.32) \\
Y_{ij} &= \begin{cases} 0, \text{if } t_j < t_i \,, \\ 1, \text{otherwise} \,. \end{cases} , && (3.33) \\
\delta_i &= \begin{cases} 0, \text{if event did not occur at time} t_i \,, \\ 1, \text{otherwise} \,. \end{cases} , && (3.34)
\end{aligned}
$$

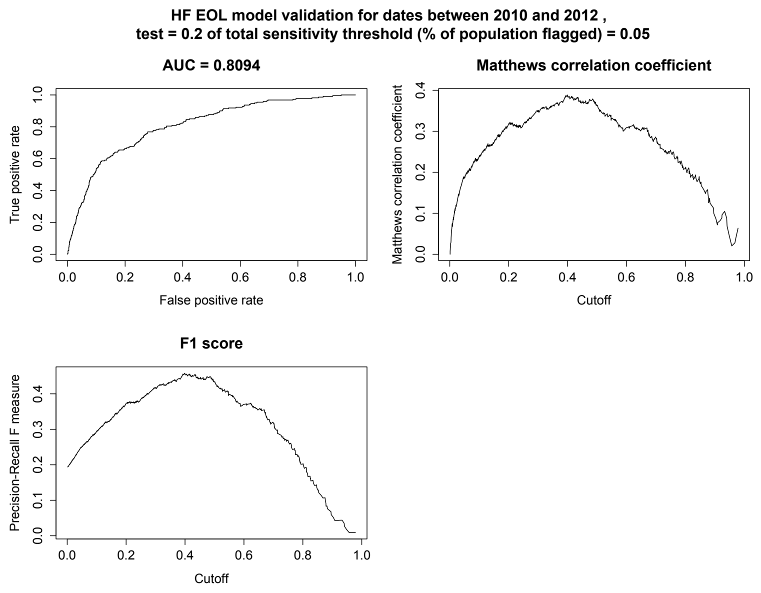
A widely accepted standard for survival analysis in R is the `survival` package [7]; a noteworthy extension of it that takes into account relative survival probability is `relsurv` [26].

Two important variables to consider in survival analysis are time since first diagnosis and age. The former reflects the individual's "lifetime" measured with respect to others with the same group of conditions, the latter relates his or her expected risk of experiencing a negative outcome to that of the general population.

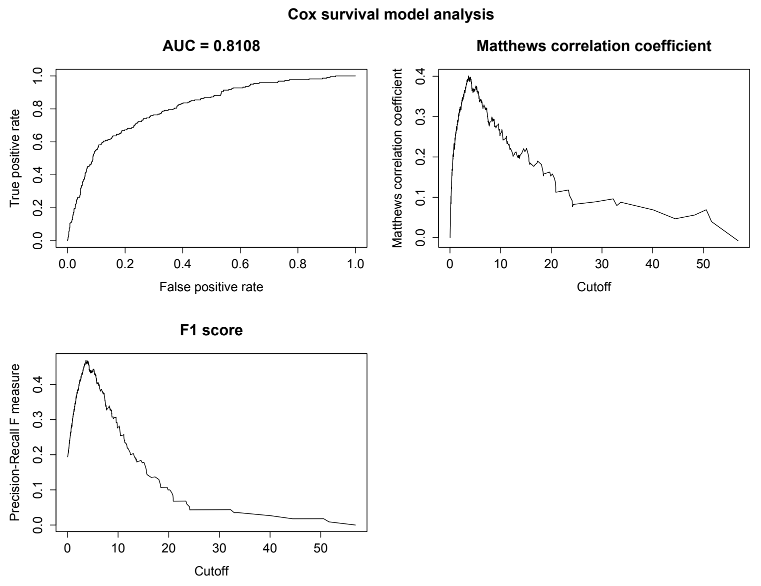
It is important to distinguish survival analysis, which is characterized by an impenetrable boundary between the sets with null and eventful outcomes, from renewal analysis where such boundary can be crossed. Clearly, the transition from alive to deceased can occur only once whereas the transition between healthy and ill can occur multiple times. Renewal analysis is governed by a similar set of equations but is conceptually different from survival analysis.

An illustrative performance comparison between a regular logistic regression model and Cox proportional hazard model used for predicting one-year mortality among heart failure patients is presented in Fig. 3.6

[1]Hence the term "semiparametric".



(a) Performance of a regular logistic regression model.



(b) Performance of a Cox proportional hazard model.

Figure 3.6: Example: Comparison of logistic regression and Cox proportional hazard model performance for predicting one-year mortality among heart failure patients.

As can be seen from Fig. 3.6, the AUC for model in question is approximately 0.81. The attained maxima are approximately 0.45 for F_1 score and 0.4 for Matthews' correlation coefficient, however, those maxima are attained at approximately 45% of the total population for the logistic regression model and 5% for the Cox proportional hazard model. This leads us to believe that, in this particular case, the latter achieves optimal accuracy for smaller population samples than the former, however, overall model accuracies is virtually identical.

The R code for performing predictive modeling in the example above can be found in Appendix D.

3.5 Data scrubbing

Several methods are available for the *imputation* of missing values [25]. A summary of currently used methods is presented in Table 3.3. For example,

Variable type	Example	Method of imputation	Comment
Indicator variable	cancer indicator	negative indicator (0)	no outcome of interest is assumed if no indicator data is present, e.g., if no cancer indicator is set, the patient is assumed to be cancer-free
Categorical variable	ethnicity	other	the missing descriptor is set to an "all-catching" category
Interval variable	binned level of income	mode	assume the most frequent occurrence for all missing data
Ordinal variable	education level	median	median works well for skewed (integer-valued) distributions
Continuous variable	systolic blood pressure	mean	we generally prefer an unbiased estimate
Date	first diagnosis date	latest date indicating contact	e.g., last visit *or* pacemaker installation date or last cardiologist contact date

Table 3.3: Missing data imputation methods

in the Heart Failure End-of-life project, the following fallback sequence was used to backfill missing first diagnosis dates:

i most recent Pacemaker Date,

ii first AICD date,

iii most recent ejection fraction measurement date,

iv most recent cardiologist visit date,

v most recent contact date.

3.6 Outliers

Outliers in the input data can be detected by examining the distribution of each independent variable. The following algorithm is suggested for detecting and eliminating outliers:

I sort the values of a predictor variable in the ascending or descending order, depending on the nature of the variable;

II eliminate obvious outliers, e.g., negative costs or 1000 mmHg blood pressure, by setting them to a predetermined fixed value (e.g., 0) or a specified aggregate statistic of the distribution (e.g., median value);

III plot the histogram of the distribution and visually inspect it;

IV if the parametric form of the distribution is known or can be inferred from theoretical or practical considerations, attempt to fit the distribution to its hypothesized shape and purge the ”tails” (can be done for either normal or non-normal cases).

V truncate the distribution if necessary (this should be considered the last resort);

3.7 Predictive variable selection

One of the most essential steps in developing a robust and accurate predictive model is variable selection. It is not uncommon to start this process with a candidate list of several hundred candidate predictors, eventually whittling it down to 10-20. While some sources advocate automated variable selection using, e.g., their significance levels, others point out that ”...a purely statistical solution is unrealistic. The role of scientific judgment cannot be overlooked.” [2]; see also [1]. Considering that it may be difficult to implement a manual solution when working with a particularly large number of variables, an automated process, e.g., backward selection, may be used to augment but not supplant the researcher's judgment; a standard R package, `caret`, is widely accepted for this purpose [16]. An algorithm for this process is outlined in Fig 3.7.

3.7.1 Removing highly correlated variables

Model coefficients β for linear (3.10) and logistic regressions (3.18) are computed numerically and are thus susceptible to stability problems if the *condition number* of the corresponding linear system is large [14], [33]. The condition number of a matrix is computed as

$$\kappa(X) = \|X\|\|X^{-1}\| \tag{3.35}$$

for a non-singular (square) matrix and as[1]

$$\begin{aligned} \kappa(X) &= \|X\|\|X^{\dagger}\| \,, && (3.36) \\ X^{\dagger} &= \begin{cases} (A^TA)^{-1}A^T, & \text{if } |A^TA| \neq 0 \,, \\ A^T(AA^T)^{-1}, & \text{if } |AA^T| \neq 0 \,. \end{cases} && (3.37) \end{aligned}$$

We can see from (3.35) that a numerically singular matrix for which $|A^TA| \approx 0$ would lead to a numerically unstable set of coefficients β with respect to a small perturbation of X:

$$\frac{\|\Delta\beta\|}{\|\beta+\Delta\beta\|} = \kappa(X)\frac{\|X\|}{\|X+\Delta X\|} \,, \tag{3.38}$$

In view of this, it is advisable to pare down highly correlated vectors as illustrated by the example below.

In the course of assessing the probability of hospitalization for chronic obstructive pulmonary disorder (COPD) patients, practitioners suggested an initial set of variables presented in Table 3.4 as candidates for inclusion in the predictive linear regression model[2].

[1]In application to (3.10) and (3.18), we are only concerned with the top line of (3.37).

[2]The full nomenclature of input variables including their type and meaning can be found in Table 5.1

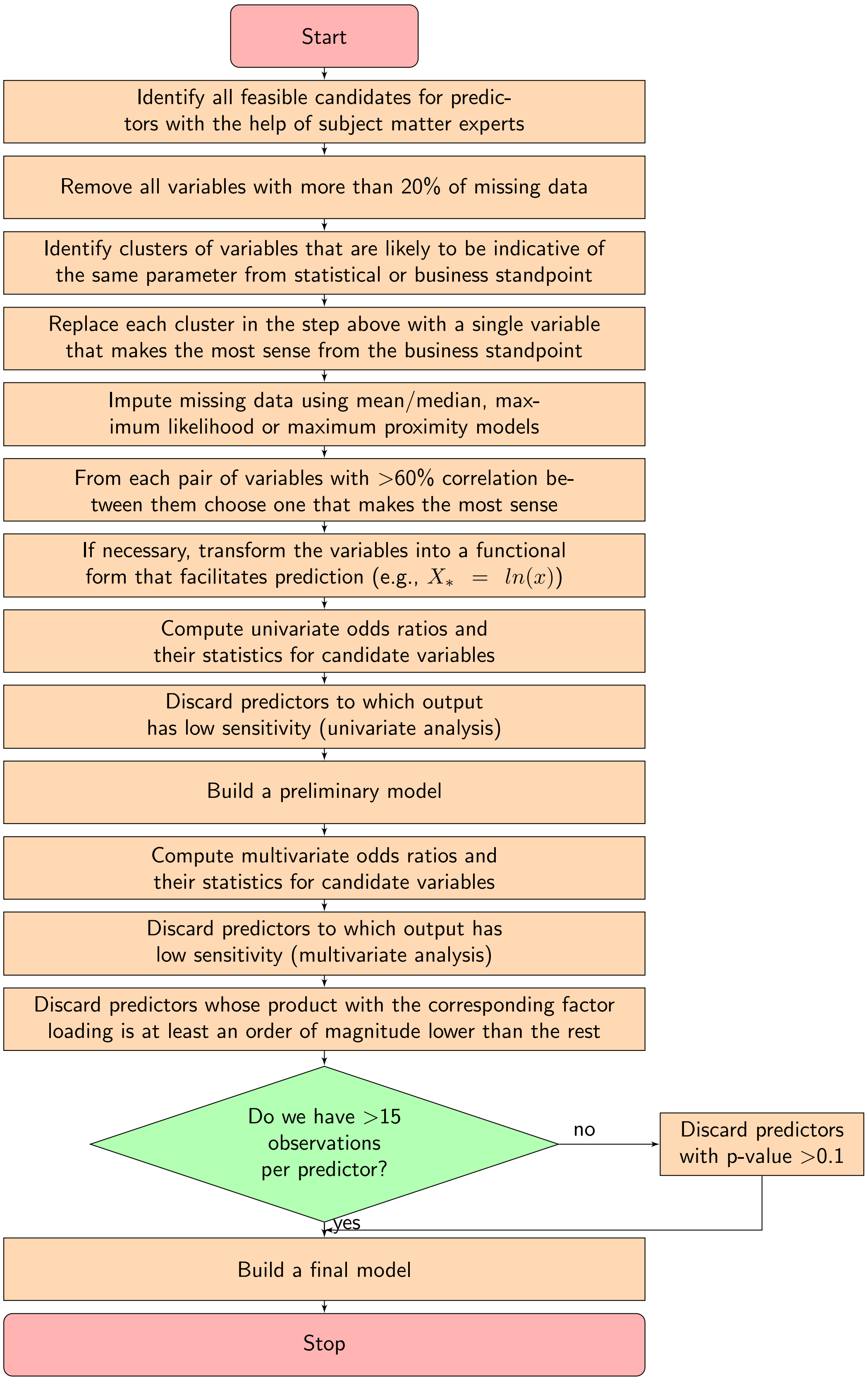

Start
Identify all feasible candidates for predictors with the help of subject matter experts
Remove all variables with more than 20% of missing data
Identify clusters of variables that are likely to be indicative of the same parameter from statistical or business standpoint
Replace each cluster in the step above with a single variable that makes the most sense from the business standpoint
Impute missing data using mean/median, maximum likelihood or maximum proximity models
From each pair of variables with >60% correlation between them choose one that makes the most sense
If necessary, transform the variables into a functional form that facilitates prediction (e.g., $X_* = ln(x)$)
Compute univariate odds ratios and their statistics for candidate variables
Discard predictors to which output has low sensitivity (univariate analysis)
Build a preliminary model
Compute multivariate odds ratios and their statistics for candidate variables
Discard predictors to which output has low sensitivity (multivariate analysis)
Discard predictors whose product with the corresponding factor loading is at least an order of magnitude lower than the rest
Do we have >15 observations per predictor?
no
Discard predictors with p-value >0.1
yes
Build a final model
Stop

Figure 3.7: An algorithm for selecting predictive variables.

Table 3.4: Example: Variable selection for COPD logistic regression model

Predictive variable
CARDO_24MM_VISIT
DAYS_SINCE_LAST_EF
DISCHARGED_DISP_30_DAYS
DISCHARGED_DISP_31_365_DAYS
DISCHARGED_DISP_365_DAYS
EJFR_NUM
HOSP_12M_VISIT
HOSP_30D_VISIT
NOT_PRN_MED_TOTAL_PRESCRIBED
NUM_ER_VISITS_30_DAYS_COPD
NUM_ER_VISITS_30_DAYS_PNEU
NUM_ER_VISITS_31_365_DAYS_COPD
NUM_ER_VISITS_31_365_DAYS_PNEU
NUM_ER_VISITS_365_DAYS_COPD
NUM_ER_VISITS_365_DAYS_PNEU
NUM_HF_HOSP_365_31_DAYS_COUNT
NUM_HOSP_30_DAYS_COPD
NUM_HOSP_30_DAYS_PNEU
NUM_HOSP_31_365_DAYS_COPD
NUM_HOSP_31_365_DAYS_PNEU
NUM_HOSP_365_DAYS_COPD
NUM_HOSP_365_DAYS_PNEU
NUM_MISSED_APPTS_365
NUM_UNIQUE_SPECIALTIES
PAT_AGE_YRS
PULMO_24MM_VISIT
TOTAL_LOS_HOSP_DAYS_LST_12_MO
TOTAL_MEDS_PRESCRIBED
ACE_INHIBITOR_PRESCRIBED
ANTICOAG_PRESCRIBED
ARB_PRESCRIBED
ASPIRIN_PLAVIX_PRESCRIBED
ASPIRIN_PRESCRIBED
BETA_BLOCKER_PRESCRIBED
CAD_IND

Continued on next page

Table 3.4 – *Continued from previous page*

Predictive variable
CALCCHANBLOCKER_PRESCRIBED
CANCER_IND
COMBOANTHYP_PRESCRIBED
DEM_IND
DIAB_IND
DIGOXIN_PRESCRIBED
EJFR_IND
ER_VISITS_365_DAYS_COPD_IND
FEMALE_IND
HF_IND
HOSP_365_DAYS_COPD_IND
HTN_IND
INSULIN_PRESCRIBED
INTUB_GT_OR_E_2DAYS_LST_2YRS
LOOPDIURETIC_PRESCRIBED
LOS_GTE_10_DAYS_HOSP
METOLAZONE_PRESCRIBED
MG_PCP_18M_SEEN_IND
MG_PCP_24M_SEEN_IND
MILDOBDOP_PRESCRIBED
MI_IND
NONINSULIN_DIAB_PRESCRIBED
O2_IND
ORALNITRATE_PRESCRIBED
ORALVASODILSPERF_PRESCRIBED
PLAVIX_PRESCRIBED
PL_ADHD_IND
PL_BIPOLAR_IND
PL_CKD_IND
PL_DEPR_IND
PL_HEPCIRR_IND
PL_HIV_IND
PL_PD_IND
PL_PM_IND
PL_PSYCH_IND
PL_SCIATICA_IND

Continued on next page

Table 3.4 – *Continued from previous page*

Predictive variable
PL_SICKLE_IND
SMOKER_IND
SPIRON_PRESCRIBED
STATIN_PRESCRIBED
THIAZIDEDIUR_PRESCRIBED

Available data included patient admission data for years 2010 to 2013[1]. The model was trained on a random subsample consisting of 80% of patient admissions from 2010 to 2013 and tested on the remaining 20% of the data. Correlation matrices for binary and continuous/interval variables are presented in Fig. 3.8 and 3.9. Specifically, highly correlated variables in binary and continuous/interval subspaces are listed in Tables 3.5 and 3.6. We se-

Predictive variable 1	Predictive variable 2	Corre-lation
HF_IND	LOOPDIURETIC_PRESCRIBED	0.6783
MG_PCP_18M_SEEN_IND	MG_PCP_24M_SEEN_IND	0.8274

Table 3.5: COPD model: Highly correlated binary variables

lect HF_IND as the more reliable and transparent of the two indicators and MG_PCP_24M_SEEN_IND as the standard medical group indicator from Table 3.5 [2] . Selecting variables from Table 3.6 is based on common sense business considerations and results in the following set: TOTAL_MEDS_-PRESCRIBED, EJFR_NUM, NUM_HOSP_365_DAYS_COPD, NUM_ER_-VISITS_365_DAYS_COPD, NUM_HOSP_365_DAYS_PNEU and NUM_ER_-VISITS_365_DAYS_PNEU. Upon comparing the resulting variable sets with the initial candidate pool in Table 3.4, we can eliminate EJFR_NUM, NUM_-HOSP_365_DAYS_COPD and NUM_ER_VISITS_365_DAYS_COPD in favor of indicators EJFR_IND, HOSP_365_DAYS_COPD_IND and ER_VISITS_-365_DAYS_COPD_IND respectively. Further analysis shows no highly correlated variables on the combined set as shown in Fig. 3.10[3]. Graphs in Fig.

[1]The original input data contains over 27,000 rows and is too voluminous to present in this document.

[2]We use this indicator frequently in our reports, as it is our locally accepted definition of a "medical froup primary care patient".

[3]Technically, Pearson correlation between numeric and indicator variables is not very informative but we present it here anyway for illustration purposes.

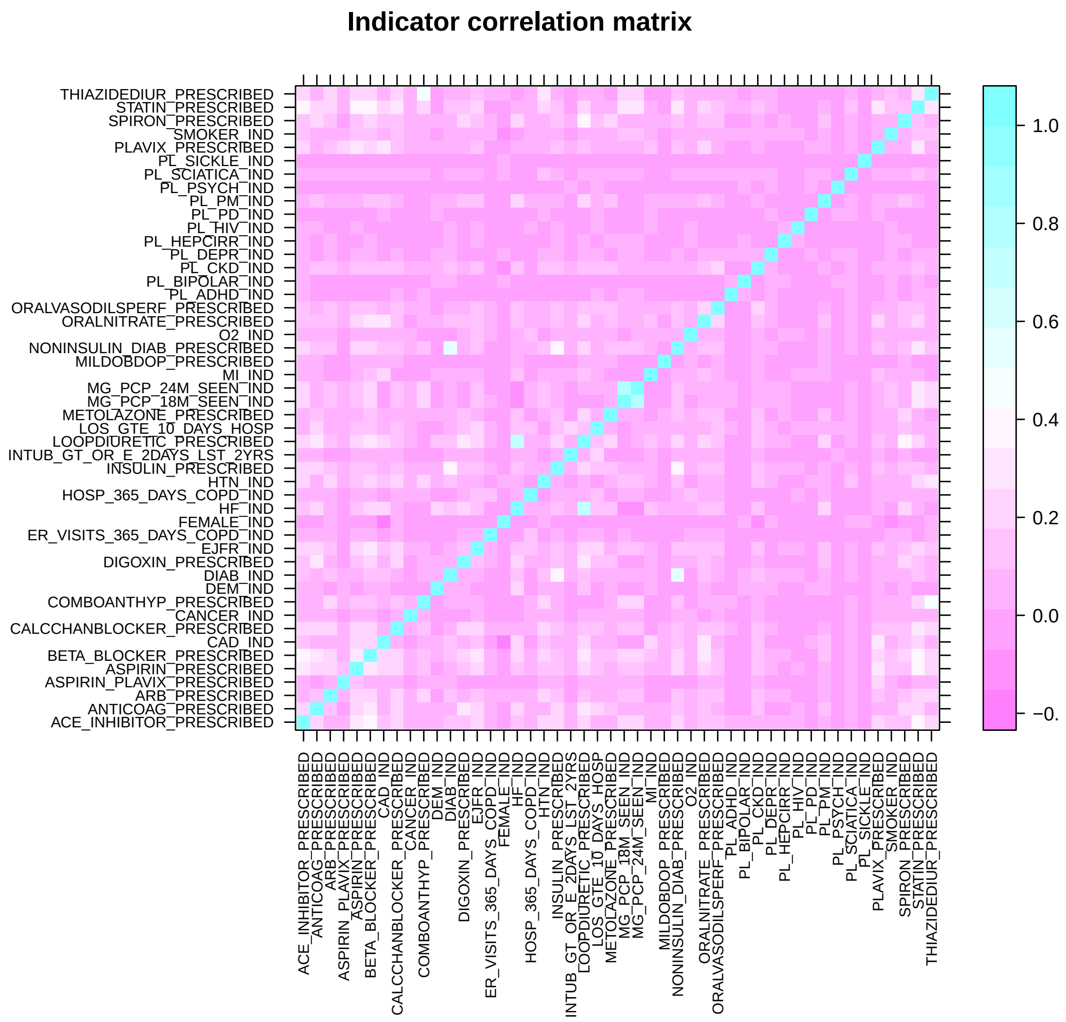


Figure 3.8: COPD Example: correlation matrix of binary, interval and continuous predictive variables

3.8 - 3.10 were generated in R using the following command:

```
        library(lattice)
        levelplot(cor(dataSet), scales=list(x=list(rot
           =90), cex=0.5))
```

Listing 3.5: Example: R code for plotting a covariance matrix.

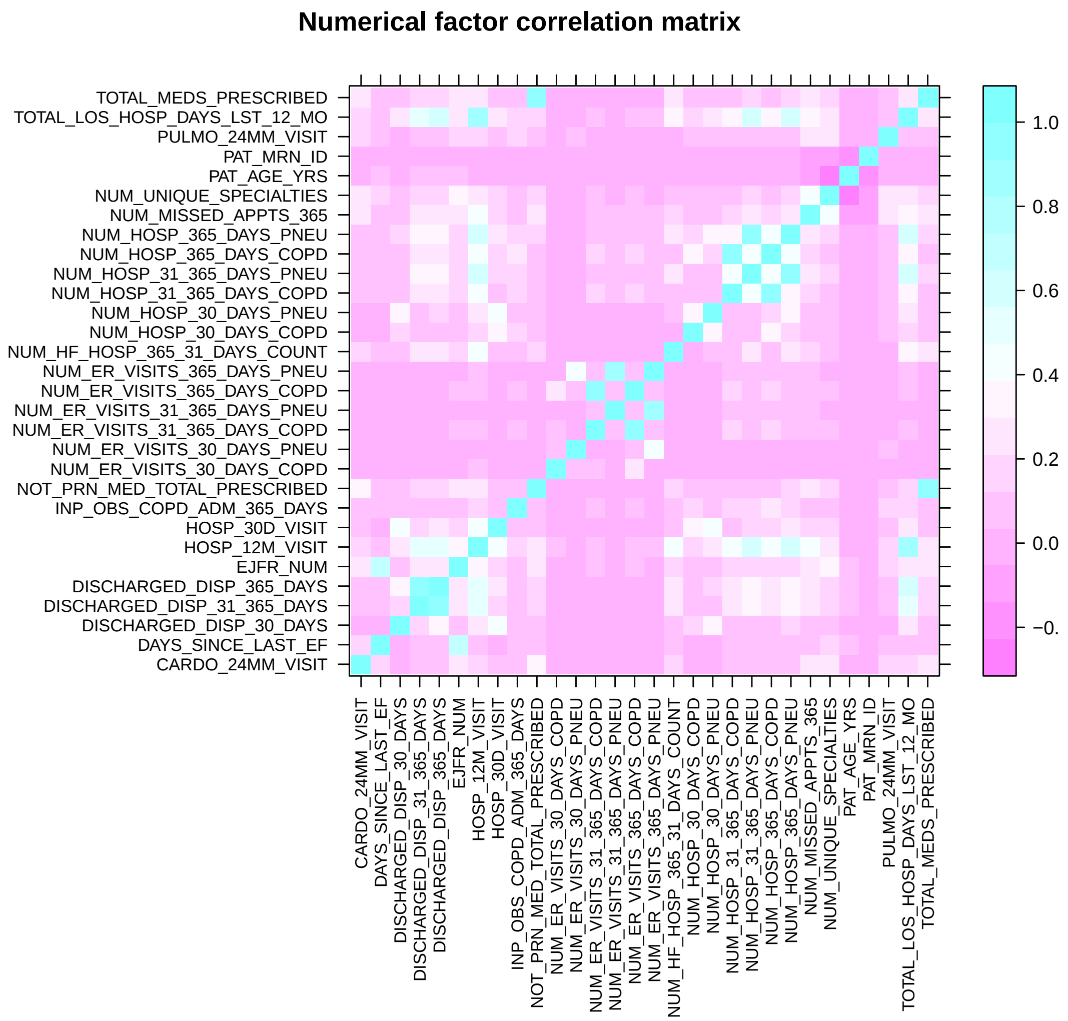


Figure 3.9: COPD Example: correlation matrix of continuous/interval predictive variables

where `dataSet` is the `dataframe` containing original input data.

3.7.2 Computing a univariate odds ratio

Consider a 2×2 *contingency table* relating predicted and actual outcomes of interest as displayed in Table 3.7. An *odds ratio* is the ratio of odds of a patient having a binary outcome of interest (1) conditional upon having the

Predictive variable 1	Predictive variable 2	Correlation
TOTAL_MEDS_-PRESCRIBED	NOT_PRN_MED_TOTAL_-PRESCRIBED	0.9924
EJFR_NUM	DAYS_SINCE_LAST_EF	0.6989
DISCHARGED_DISP_365_-DAYS	DISCHARGED_DISP_31_365_DAYS	0.9575
NUM_HOSP_31_365_DAYS_-COPD	NUM_HOSP_365_DAYS_COPD	0.9605
NUM_ER_VISITS_31_365_-DAYS_COPD	NUM_ER_VISITS_365_DAYS_COPD	0.9656
NUM_HOSP_31_365_DAYS_-PNEU	NUM_HOSP_365_DAYS_PNEU	0.9643
NUM_ER_VISITS_31_365_-DAYS_PNEU	NUM_ER_VISITS_365_DAYS_PNEU	0.9128
HOSP_12M_VISIT	TOTAL_LOS_HOSP_DAYS_LST_12_-MO	0.8319
NUM_HOSP_31_365_DAYS_-PNEU	TOTAL_LOS_HOSP_DAYS_LST_12_-MO	0.6173
NUM_HOSP_365_DAYS_-PNEU	TOTAL_LOS_HOSP_DAYS_LST_12_-MO	0.6338

Table 3.6: COPD model: Highly correlated continuous/interval variables

Predictive variable	Outcome		Total
	1	0	
1	n_{11}	n_{10}	n_{1*}
0	n_{01}	n_{00}	n_{2*}
Total	n_{*1}	n_{*2}	N

Table 3.7: Predictive variable and outcome of interest

property described by the predictive variable to the odds of the patient not having an outcome of interest (0) conditional upon not having that property:

$$OR_{uni}^{(ind)} = \frac{\frac{n_{11}}{n_{10}}}{\frac{n_{01}}{n_{00}}} = \frac{n_{11}n_{00}}{n_{10}n_{01}} \tag{3.39}$$

Statistically, an odds ratio describes how much more likely the patient is to have an outcome of interest if he possesses a property thought to be predictive of the outcome compared to not having that property. If the

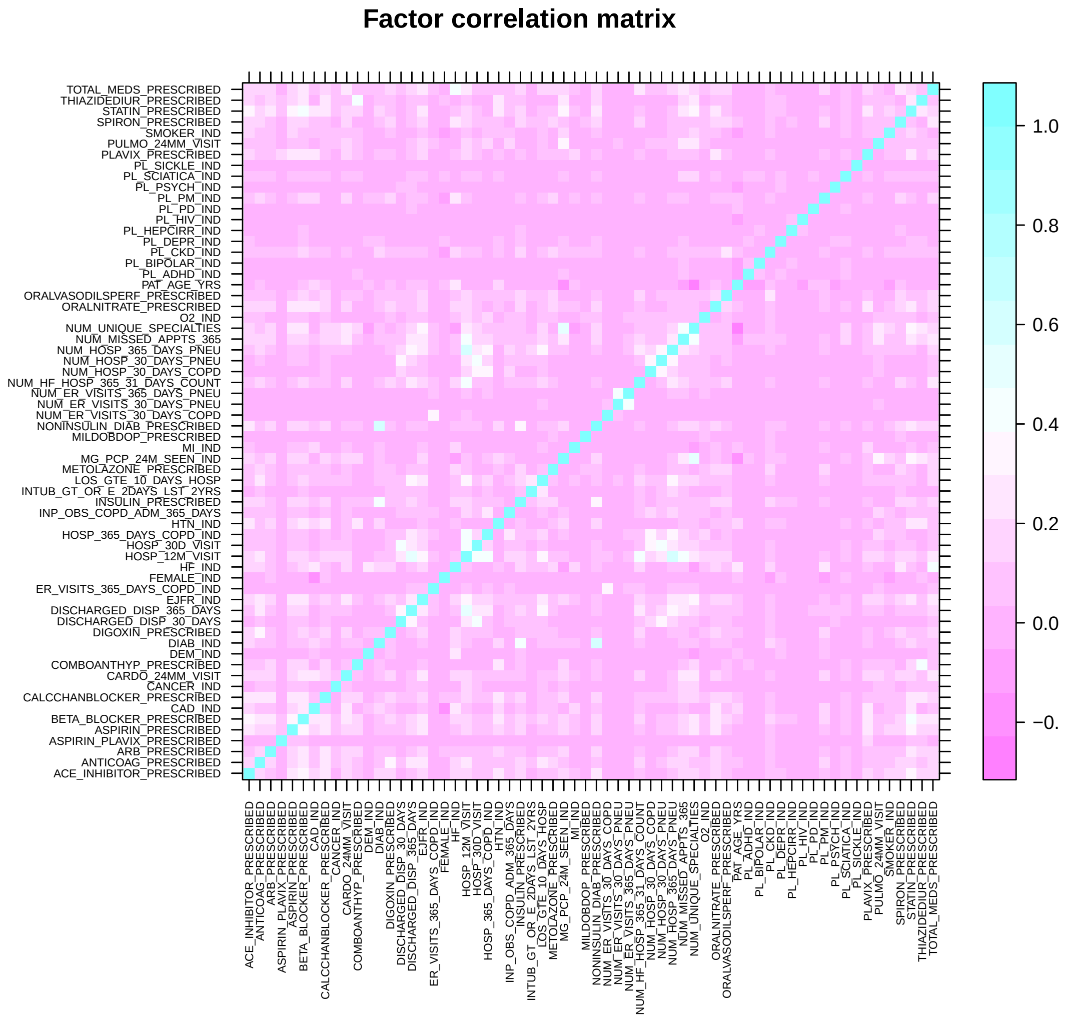


Figure 3.10: COPD Example: correlation matrix of binary predictive variables

odds ratio or its inverse are different from 1, then there is a chance that the candidate predictive variable indeed possesses predictive power. This hypothesis can be statistically justified if the confidence interval for the odds ratio does not include 1 at the significance level α.

An example of a contingency matrix for hospital admissions of heart failure patients contingent upon them having had an ejection fraction test previously ordered is presented in Table 3.8. Here the corresponding ratio

Ejection fraction ever ordered?	Hospitalized in the following 365 days?		Total
	yes	no	
yes	5	90	95
no	20	1000	1020
TOTAL	25	1090	1115

Table 3.8: Example: Predictive variable and outcome of interest

is

$$OR^{(ind)}_{uni} = \frac{\frac{5}{90}}{\frac{20}{1000}} = \frac{1}{18 \times 0.02} = 2.78 \ ,$$

signifying a potentially high predictive value of ejection fraction having been ordered in the past when forecasting future hospitalizations within the following year.

If the predictor variable under consideration is categorical with more than two levels rather than indicator type, a 2×2 contingency table cannot be constructed and (3.39) does not apply. In this case, either of the following modification of the algorithm for calculating the odds ratio can be employed to calculate a suitable proxy:

I *One-vs.-the-rest*:

(a) compute the proportion of the total population that belongs to each category;

(b) roll up categories containing the percentage of the population that is smaller than a predetermined lower boundary (e.g., 5%);

(c) calculate the number of positive and negative outcomes of interest for the remainder of the population excluding each (rolled-up) category in turn;

(d) construct the 2×2 contingency table as before and compute the "one-vs.the-rest" odds ratio for each category following the algorithm for indicator variables described above.

II *Benchmark*:

(a) roll up sparsely populated categories as described above;

(b) select a "benchmark" category that makes business sense (e.g., "married" if examining marital status); in many instances, it makes sense to choose the most populous category as the benchmark;

(c) for each category, construct the 2×2 contingency table against the benchmark and compute the "benchmark" odds ratio as you would for an indicator variable.

An example of the *one-vs.-the-rest* algorithm is given by blood utilization data presented in Table 3.9. Since there are no sparse categories, i.e., the ones containing less than 5% of the total population, we can separate each hospital (pavilion) in turn from the rest and generate 2×2 contingency tables as shown in Table 3.10. Judging by the odds ratios presented in Table 3.10,

Pavilion	# of patients		Total	% of Grand Total
	transfused	not transfused		
A	800	24,200	25,000	40.32
B	800	12,200	13,000	20.97
C	700	13,300	14,000	22.58
D	700	9,300	10,000	16.13
TOTAL	3,000	59,000	62,000	100.00

Table 3.9: Example: Blood utilization data for building *one-vs.the-rest* contingency table[1]

Pavilion	# of patients		Odds	Odds ratio	CI lower	CI upper
	transfused	not transfused				
A	800	24,200	0.033	0.52	0.48	0.57
Other	2,200	34,800	0.063			
B	800	12,200	0.066	1.39	1.28	1.52
Other	2,200	46,800	0.047			
C	700	13,300	0.053	1.05	0.96	1.14
Other	2,300	45,700	0.050			
D	700	9,300	0.075	1.62	1.49	1.78
Other	2,300	49,700	0.046			

Table 3.10: Example: *One-vs.the-rest* contingency tables by hospital.

[1]The total adds up to 100% within the roundoff error.

Marital status	Hospitalized in the following 365 days?		Total	% of Grand Total
	yes	no		
Divorced	58	2,220	2,278	8.33
Engaged	0	8	8	0.03
Legally Separated	1	39	40	0.15
Life Partner	2	34	36	0.13
Married	305	12,160	12,465	45.61
Separated (Not Legally)	1	96	97	0.35
Single	112	3,671	3783	13.84
Unknown	2	352	354	1.30
Widowed	251	8,020	8,271	30.26
TOTAL	732	26,600	27,332	100.00

Table 3.11: Example: Predictive variable and outcome of interest

Pavilion C is the only hospital whose identity appears to have no discernible "predictive" influence on the number of blood transfusions compared to the rest of the pavilions.

As an example of the *benchmark* algorithm, consider admission data presented in Table 3.11. We now roll up sparse categories, e.g., the ones containing less than 5% of the total population, by merging "Engaged", "Legally Separated", "Life Partner" and "Separated (Not Legally)" into category "Other" as shown in Table 3.12. The most populous category, "Married", is a natural benchmark selection against which the odds ratios and their statistics can be computed. An example for category "Divorced" in shown in Table 3.13. Here the odds ratio is

$$OR = \frac{\frac{58}{2{,}220}}{\frac{305}{12{,}160}} = \frac{0.0261}{0.0251} = 1.04\ ,$$

for the remaining categories as shown in Table 3.14 A straightforward argument based on the data in Table 3.14 would favor "Widowed" as a predictor of hospitalizations since its odds ratio is statistically significantly different from 1 at the 99% level ($p = 0.01$) and its confidence interval (CI) does not include 1 at the 95% significance level ($p = 0.05$).

If the predictor variable is continuous rather than categorical, it could conceivably be transformed into the categorical form by "binning" its values into intervals, however, this approach is generally not recommended.

Marital status	Hospitalized in the following 365 days?		Total	% of Grand Total
	yes	no		
Divorced	58	2,220	2,278	8.33
Married	305	12,160	12,465	45.61
Single	112	3,671	3783	13.84
Other	6	529	535	1.96
Widowed	251	8,020	8,271	30.26
TOTAL	732	26,600	27,332	100.00

Table 3.12: Example: Predictive variable and outcome of interest, rolled-up "Marital Status"

Marital status	Hospitalized in the following 365 days?		Total
	yes	no	
Divorced	58	2,220	2,278
Married	305	12,160	12,465
TOTAL	363	14,380	14,743

Table 3.13: Example: Odds ratio for "Divorced" vs. "Married".

Marital status	Odds Ratio	CI Lower	CI Upper	p-value
Divorced	1.04	0.78	1.38	0.779
Single	1.21	0.98	1.52	0.081
Other	0.45	0.20	1.02	0.056
Widowed	1.25	1.05	1.48	0.010

Table 3.14: Example: *Benchmark* odds ratios and their statistics.

Instead, an "incremental" odds ratio is computed as follows:

1. construct a univariate logistic regression model for the variable in question as
$$\ln\left(\frac{p(x)}{1-p(x)}\right) = b_0 + ax \;, \tag{3.40}$$
where b_0 is the intercept of the logistic equation, a is the slope of the (univariate logistic regression) line;

2. observe that
$$\ln\left(\frac{p(x+1)}{1-p(x+1)}\right) = b_0 + a(x+1) \;, \tag{3.41}$$
and hence
$$\begin{aligned}&\ln\left(\frac{p(x+1)}{1-p(x+1)}\right) - \ln\left(\frac{p(x)}{1-p(x)}\right) = \\ &\ln\left(\frac{p(x+1)[1-p(x)]}{p(x)[1-p(x+1)]}\right) = a = \ln\left(OR_{uni}^{cont}\right) \;.\end{aligned} \tag{3.42}$$

Exponentiating both sides, we obtain
$$OR_{uni}^{cont} = e^{a} \;. \tag{3.43}$$

The odds ratio defined by (3.43) can be viewed as a proportional increase in the odds of encountering an outcome of interest corresponding to a unitary increase in the value of the (continuous) predictive variable of interest. Note here that (3.43) makes sense only if the predictive variable can indeed vary by 1, if not, it needs to be reformulated with respect to the permissible increment δ [15]:

$$\begin{aligned}&\ln\left(\frac{p(x+\delta)}{1-p(x+\delta)}\right) - \ln\left(\frac{p(x)}{1-p(x)}\right) = \\ &\ln\left(\frac{p(x+\delta)[1-p(x)]}{p(x)[1-p(x+\delta)]}\right) = \delta a \;,\end{aligned} \tag{3.44}$$
$$\ln\left(OR(\delta)_{uni}^{cont}\right) = \delta a \;, \tag{3.45}$$
$$OR(\delta)_{uni}^{cont} = e^{\delta a} \;. \tag{3.46}$$

An instructive example of the foregoing is the "incremental" odds ratio with respect to patient age as described in Table 3.15. We construct a univariate

Age	Hospitalized in the following 365 days?		Age	Hospitalized in the following 365 days?		Age	Hospitalized in the following 365 days?		Age	Hospitalized in the following 365 days?	
	yes	no		yes	no		yes	no		yes	no
0	0	0	27	0	8	54	4	175	81	28	889
1	0	0	28	0	7	55	5	202	82	39	927
2	0	0	29	0	10	56	5	233	83	39	969
3	0	0	30	0	16	57	9	247	84	26	984
4	0	0	31	0	16	58	9	257	85	38	916
5	0	0	32	0	17	59	5	284	86	27	908
6	0	7	33	0	11	60	10	302	87	25	830
7	0	11	34	0	8	61	10	321	88	24	779
8	0	10	35	0	8	62	5	347	89	19	749
9	0	12	36	0	9	63	9	413	90	17	642
10	0	13	37	0	15	64	3	429	91	17	586
11	0	11	38	0	17	65	14	442	92	15	544
12	0	12	39	0	25	66	12	493	93	6	438
13	0	10	40	0	31	67	10	470	94	3	389
14	0	7	41	0	30	68	10	541	95	3	332
15	0	6	42	0	32	69	14	561	96	4	258
16	0	6	43	0	33	70	17	606	97	3	224
17	0	8	44	0	41	71	21	596	98	2	183
18	0	13	45	0	46	72	18	592	99	2	133
19	0	14	46	0	47	73	21	596	100	0	97
20	0	12	47	0	71	74	26	636	101	0	64
21	0	13	48	0	77	75	21	679	102	0	40
22	0	11	49	0	98	76	20	700	103	0	38
23	0	11	50	4	120	77	23	737	104	0	30
24	0	11	51	3	121	78	24	747	105	0	18
25	0	8	52	5	143	79	25	820	106	0	9
26	0	7	53	3	165	80	31	838			

Table 3.15: Example: Patient outcome by age.

logistic regression model from the data in Table 3.15 using (3.40 - 3.43).

$$\ln\left(\frac{p(x)}{1-p(x)}\right) = -19.58 + 0.0247x \ , \tag{3.47}$$

and, therefore,

$$OR_{uni}^{cont} = e^{0}.0247 = 1.025 \ . \tag{3.48}$$

The odds ratio in (3.48) does not reveal much of a pattern of dependency of the probability of hospitalization on the patient's age. Alternatively, considering an increment of 10 years instead, we obtain:

$$OR(10)_{uni}^{cont} = e^{0.247} = 1.28 \ , \tag{3.49}$$

and thus the odds ratio over a 10-year interval appears[1] to have more potential predictive power than its conventional counterpart defined by (3.48). Regardless of the size of the increment δ, the graph of $logit(p(x))$ in Fig. 3.11 leads one to be skeptical about the influence of age as a continuous variable on the likelihood of hospitalization. Its inclusion in the final set of variables needs to be justified by examining overall model performance as described in section 3.10.2. The R code used for generating Fig. 3.11 is presented in Listing 3.6.

In the example in Section 3.7.1, the odds ratio matrix computed for binary variables is presented in Table 3.16[2]

Table 3.16: Example: Univariate odds ratio statistics for COPD logistic regression model - binary and indicator variables.

Variable	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary	
ACE_INHIBITOR_-PRESCRIBED	1.8738	0.0000	1.5894	2.2090	*

Continued on next page

[1] Here we are not considering the confidence interval of $OR(10)_{uni}^{cont}$

[2] Table 3.16 was generated by calling `odds.ratio.save` from package `NS.CA.modelUtils`, see Appendix E.4; function signature can be found, e.g., in Listing D in Appendix D.

Table 3.16 – *Continued from previous page*

Variable	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary	
ANTICOAG_-PRESCRIBED	2.1877	0.0000	1.8446	2.5946	*
ARB_PRESCRIBED	2.0724	0.0000	1.7051	2.5187	*
ASPIRIN_PLAVIX_-PRESCRIBED	1.6495	0.3972	0.5179	5.2541	
ASPIRIN_PRESCRIBED	1.6309	0.0000	1.3749	1.9347	*
BETA_BLOCKER_-PRESCRIBED	1.6672	0.0000	1.4203	1.9569	*
CAD_IND	0.8656	0.1490	0.7115	1.0530	
CALCCHANBLOCKER_-PRESCRIBED	2.3203	0.0000	1.9747	2.7262	*
CANCER_IND	1.7982	0.0115	1.1405	2.8350	*
COMBOANTHYP_-PRESCRIBED	1.3457	0.0254	1.0372	1.7461	*
DEM_IND	0.8037	0.5692	0.3786	1.7059	
DIAB_IND	0.9554	0.6724	0.7734	1.1803	
DIGOXIN_PRESCRIBED	1.4132	0.0159	1.0670	1.8716	*
EJFR_IND	3.8428	0.0000	3.2834	4.4974	*
ER_VISITS_365_DAYS_-COPD_IND	10.4165	0.0000	7.3204	14.8222	*
FEMALE_IND	1.0764	0.3582	0.9200	1.2594	
HF_IND	0.9542	0.6393	0.7842	1.1610	
HOSP_365_DAYS_COPD_-IND	12.4944	0.0000	10.4371	14.9571	*
HTN_IND	0.9466	0.4894	0.8100	1.1061	
INSULIN_PRESCRIBED	1.6748	0.0001	1.2896	2.1749	*
INTUB_GT_OR_E_-2DAYS_LST_2YRS	1.8815	0.1693	0.7639	4.6339	
LOOPDIURETIC_-PRESCRIBED	1.4818	0.0005	1.1877	1.8489	*
LOS_GTE_10_DAYS_-HOSP	3.2674	0.0000	2.6066	4.0957	*
METOLAZONE_-PRESCRIBED	1.5546	0.0784	0.9512	2.5408	
MG_PCP_18M_SEEN_IND	2.1249	0.0000	1.8176	2.4841	*

Continued on next page

Table 3.16 – *Continued from previous page*

Variable	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary	
MI_IND	0.8766	0.5153	0.5895	1.3036	
MILDOBDOP_-PRESCRIBED	2.2253	0.5827	0.1283	38.5944	
NONINSULIN_DIAB_-PRESCRIBED	1.3872	0.0042	1.1087	1.7357	*
O2_IND	4.7988	0.0000	3.7379	6.1609	*
ORALNITRATE_-PRESCRIBED	1.3678	0.0276	1.0352	1.8071	*
ORALVASODILSPERF_-PRESCRIBED	3.0655	0.0000	2.1321	4.4075	*
PL_ADHD_IND	2.5317	0.1201	0.7847	8.1676	
PL_BIPOLAR_IND	0.5756	0.2744	0.2138	1.5498	
PL_CKD_IND	1.4633	0.0272	1.0438	2.0514	*
PL_DEPR_IND	0.9884	0.9268	0.7713	1.2667	
PL_HEPCIRR_IND	0.6641	0.2857	0.3133	1.4079	
PL_HIV_IND	2.9190	0.1449	0.6914	12.3239	
PL_PD_IND	0.5583	0.2487	0.2074	1.5030	
PL_PM_IND	1.5748	0.0039	1.1565	2.1443	*
PL_PSYCH_IND	1.5381	0.2667	0.7196	3.2877	
PL_SCIATICA_IND	1.2370	0.0267	1.0249	1.4931	*
PL_SICKLE_IND	9.4805	0.0444	1.0582	84.9378	*
PLAVIX_PRESCRIBED	1.8720	0.0000	1.4833	2.3630	*
SMOKER_IND	3.2665	0.0000	2.4959	4.2749	*
SPIRON_PRESCRIBED	1.9064	0.0000	1.4722	2.4685	*
STATIN_PRESCRIBED	1.5582	0.0000	1.3314	1.8237	*
THIAZIDEDIUR_-PRESCRIBED	1.7971	0.0000	1.4951	2.1602	*
African American : Caucasian	0.5408	0.0000	0.4098	0.7135	*
Asian : Caucasian	0.9925	0.9833	0.4892	2.0134	
Other : Caucasian	1.4645	0.0071	1.1091	1.9336	*
Divorced : Married	1.0628	0.6914	0.7868	1.4355	
Single : Married	1.2016	0.1233	0.9513	1.5177	
Unknown : Married	0.4367	0.0679	0.1795	1.0627	

Continued on next page

Table 3.16 – *Continued from previous page*

Variable	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary	
Widowed : Married	1.2698	0.0087	1.0623	1.5178	*

As can be seen from Table 3.16, the most significant predictive variables with respect to their odd ratios are HOSP_365_DAYS_COPD_IND, ER_VISITS_365_DAYS_COPD_IND, PL_SICKLE_IND and O2_IND. The confidence interval for the odds ratio of hospitalization as a function of sickle cell anemia is very wide alerting us to the possible unreliability of this variable as a predictor. Additional data (not presented here for the sake of brevity) shows that the number of patients with sickle cell anemia is too small to derive meaningful conclusions, and therefore, this variable can be dropped from consideration.

The odds ratio matrix computed for continuous / interval variables is presented in Table 3.17[1]

Table 3.17: Example: Univariate odds ratio statistics for COPD logistic regression model - interval and continuous variables.

Variable	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary	
CARDO_24MM_VISIT	1.1441	0.0000	1.1028	1.1869	*
DAYS_SINCE_LAST_EF	1.0009	0.0000	1.0007	1.0011	*
DISCHARGED_DISP_30_-DAYS	5.5997	0.0000	3.9663	7.9056	*
DISCHARGED_DISP_31_-365_DAYS	4.2532	0.0000	3.5935	5.0340	*
DISCHARGED_DISP_-365_DAYS	4.3690	0.0000	3.7120	5.1422	*

Continued on next page

[1]Table 3.17 was generated by calling `run.glm.model` from package texttttNS.CA.modelUtils, see Appendix E.4; function signature can be found, e.g., in Listing D in Appendix D.

Table 3.17 – *Continued from previous page*

Variable	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary	
EJFR_NUM	1.0234	0.0000	1.0211	1.0257	*
HOSP_12M_VISIT	1.5409	0.0000	1.4907	1.5929	*
HOSP_30D_VISIT	3.8266	0.0000	3.2682	4.4804	*
NOT_PRN_MED_-TOTAL_PRESCRIBED	1.1319	0.0000	1.1107	1.1535	*
NUM_ER_VISITS_30_-DAYS_COPD	12.0603	0.0000	5.5671	26.1269	*
NUM_ER_VISITS_30_-DAYS_PNEU	15.1914	0.0012	3.8305	60.2473	*
NUM_ER_VISITS_31_365_-DAYS_COPD	3.7188	0.0000	2.7820	4.9710	*
NUM_ER_VISITS_31_365_-DAYS_PNEU	4.2475	0.0032	1.8979	9.5060	*
NUM_ER_VISITS_365_-DAYS_COPD	3.9804	0.0000	3.0254	5.2368	*
NUM_ER_VISITS_365_-DAYS_PNEU	5.4457	0.0000	2.7264	10.8770	*
NUM_HF_HOSP_365_31_-DAYS_COUNT	1.8967	0.0000	1.6273	2.2108	*
NUM_HOSP_30_DAYS_-COPD	16.0163	0.0000	11.7692	21.7960	*
NUM_HOSP_30_DAYS_-PNEU	5.9912	0.0000	4.3968	8.1638	*
NUM_HOSP_31_365_-DAYS_COPD	4.1532	0.0000	3.6970	4.6656	*
NUM_HOSP_31_365_-DAYS_PNEU	2.5963	0.0000	2.3653	2.8498	*
NUM_HOSP_365_DAYS_-COPD	4.3934	0.0000	3.9402	4.8987	*
NUM_HOSP_365_DAYS_-PNEU	2.5560	0.0000	2.3428	2.7886	*
NUM_MISSED_APPTS_-365	1.0664	0.0000	1.0562	1.0766	*
NUM_UNIQUE_-SPECIALTIES	1.3969	0.0000	1.3541	1.4411	*
PAT_AGE_YRS	1.0031	0.3006	0.9982	1.0079	
PULMO_24MM_VISIT	1.3583	0.0000	1.3182	1.3997	*

Continued on next page

Table 3.17 – *Continued from previous page*

Variable	Odds Ratio	$Pr(\lvert z \rvert > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary	
TOTAL_LOS_HOSP_-DAYS_LST_12_MO	1.0595	0.0000	1.0537	1.0653	*
TOTAL_MEDS_-PRESCRIBED	1.1203	0.0000	1.1020	1.1388	*

As can be seen from Table 3.17, the most significant predictive variables with respect to their odd ratios are NUM_HOSP_30_DAYS_COPD, NUM_-ER_VISITS_30_DAYS_PNEU, and NUM_ER_VISITS_30_DAYS_COPD. We can also observe that PAT_AGE_YRS appears to be insignificant from the point of view of the corresponding odds ratio. In spite of this, we need to bear in mind that, as pointed out in Section 3.7.2, a one year increase in patient age does not change the odds of hospitalization significantly and thus patient's age cannot be automatically discarded from the final model.

3.7.3 Computing a multivariate odds ratio

The odds ratio defined in Section 3.7.2 loses its meaning for a multivariate model regardless of whether the predictive variables are of indicator, categorical or continuous type. Fortunately, (3.42) can be generalized to the case of a multivariate model once we realize that all terms in a multivariate logistic equation except a vanish the same way as they did in (3.42) once we construct the "incremental" odds ratio. In view of this, our algorithm will proceed as follows:

1. construct a multivariate logistic regression model for the variable in question as

$$\ln\left(\frac{p(x)}{1-p(x)}\right) = b_0 + \sum_{i=1}^{N} a_i x_i \ , i = \overline{1,N} \ , \tag{3.50}$$

where $x = (x_1, \dots, x_n)^T$ is the vector of predictive variables;

2. observe that

$$\ln\left(\frac{p(x_i+1)}{1-p(x_i+1)}\right) = b_0 + \sum_{k=1}^{i-1} a_i x_i + a_i(x_i+1) + \sum_{k=i+1}^{N} a_i x_i \ , \tag{3.51}$$

```
logit <- function( x ) { return( log( x / ( 1 - x ) )
    ) }

COPDadmRaw <- read.csv( "../Data/COPD_ALL_ALIVE.csv" )
outcome <- "INP_OBS_COPD_ADM_365_DAYS"
COPDadm <- transform( COPDadmRaw[, c( "PAT_AGE_YRS",
    outcome )],
                            PAT_AGE_YRS=round( PAT_AGE_YRS
                                ) )
COPDadm[is.na( COPDadm[, outcome] ), outcome] <- 0
COPDform <- as.formula( paste( "PAT_AGE_YRS ~",
    outcome, sep="" ) )
COPDprod <- transform( dcast(COPDadm, COPDform,
    length ) )
colnames( COPDprod ) <- c( "PAT_AGE_YRS", "No", "Yes"
    )
COPDprod <- mutate( COPDprod, Total=Yes + No,
    Prob=Yes / Total,
                        Logit=logit( Prob ) )
COPDprod[! is.finite( COPDprod$Logit ), "Logit"] <-
    -25

plot( COPDprod$PAT_AGE_YRS, COPDprod$Logit,
        main="Logit of the probability of
            hospitalization", xlab="Age, yrs.",
        ylab="Logit(p) = p / (1 - p)" )
```

Listing 3.6: Example: R code for generating the logit of age-dependent propability of hospitalization.

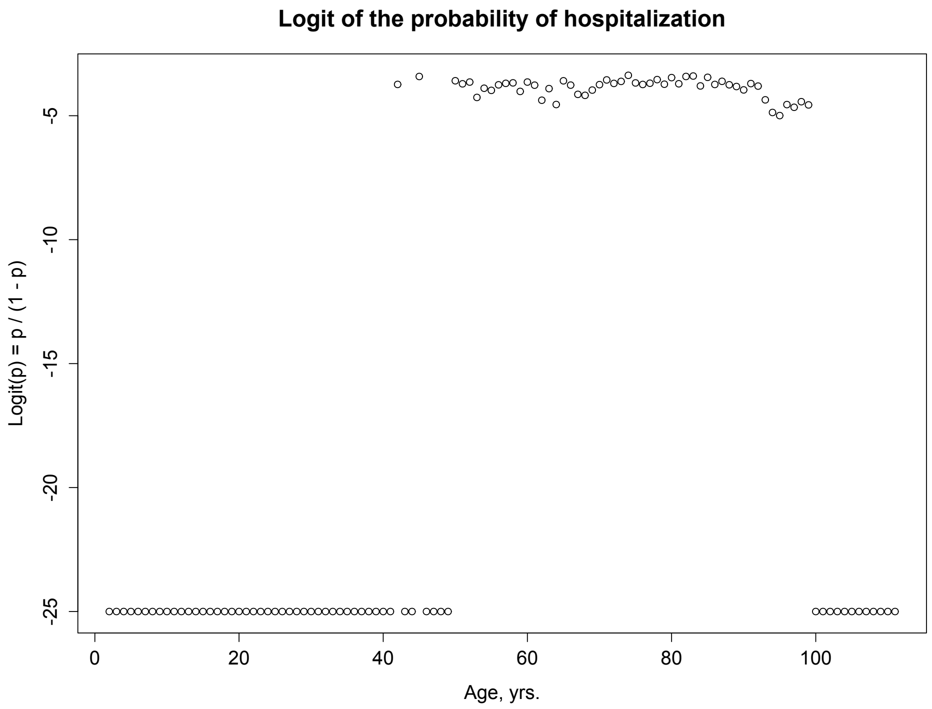


Figure 3.11: Example: Probability of hospitalization from univariate logistic regression on patient age.

and hence

$$\ln\left(\frac{p(x_i+1)}{1-p(x_i+1)}\right) - \ln\left(\frac{p(x_i)}{1-p(x_i)}\right) =$$
$$\ln\left(\frac{p(x_i+1)[1-p(x_i)]}{p(x_i)[1-p(x_i+1)]}\right) = a_i = \ln\left(OR^{cont}_{multi}\right) \ . \qquad (3.52)$$

Exponentiating both sides, we obtain

$$OR^{cont}_{multi} = e^{a_i} \ . \qquad (3.53)$$

The odds ratio defined by (3.53) represents a proportional increase in the odds of encountering an outcome of interest corresponding to a unitary increase in the value of the respective (continuous) predictive variable of interest. The same note of caution with respect to the domain of the "incremental" multivariate odds ratio applies here as in the univariate case above.

In the example in Section 3.7.1, the odds ratio matrix computed for both continuous / interval and indicator variables is presented in Table 3.18.

Table 3.18: . Example: Multivariate odds ratio statistics for COPD logistic regression model

Variable	Univariate analysis					Multivariate analysis				
	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary				Lower boundary	Upper boundary	
ACE_INHIBITOR_-PRESCRIBED	1.76	0.0000	1.49	2.08	*	1.19	0.0930	1.00	1.42	
African American : Caucasian	0.53	0.0000	0.40	0.69	*	0.00	0.0000	0.00	0.00	
ANTICOAG_-PRESCRIBED	2.10	0.0000	1.77	2.49	*	1.21	0.0832	1.01	1.45	
ARB_PRESCRIBED	2.01	0.0000	1.65	2.45	*	1.27	0.0441	1.04	1.54	*
Asian : Caucasian	0.78	0.4497	0.41	1.48		0.00	0.0000	0.00	0.00	
ASPIRIN_PLAVIX_-PRESCRIBED	2.24	0.1191	0.81	6.15		1.76	0.3036	0.71	4.37	
ASPIRIN_PRESCRIBED	1.75	0.0000	1.48	2.07	*	0.91	0.3573	0.76	1.08	
BETA_BLOCKER_-PRESCRIBED	1.73	0.0000	1.47	2.03	*	1.03	0.7966	0.86	1.22	
CAD_IND	0.88	0.1872	0.72	1.07		0.78	0.0505	0.64	0.96	
CALCCHANBLOCKER_-PRESCRIBED	2.29	0.0000	1.95	2.69	*	1.25	0.0298	1.06	1.48	*
CANCER_IND	1.95	0.0027	1.26	3.01	*	0.75	0.2322	0.50	1.12	
CARDO_24MM_VISIT	1.13	0.0000	1.09	1.17	*	0.95	0.0718	0.90	1.00	

Continued on next page

Table 3.18 – *Continued from previous page*

Variable	Univariate analysis					Multivariate analysis				
	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary				Lower boundary	Upper boundary	
COMBOANTHYP_PRESCRIBED	1.41	0.0090	1.09	1.82	*	0.98	0.8985	0.76	1.27	
DAYS_SINCE_LAST_EF	1.00	0.0000	1.00	1.00	*	0.00	0.0000	0.00	0.00	
DEM_IND	0.69	0.3698	0.31	1.55		1.10	0.8376	0.52	2.33	
DIAB_IND	0.97	0.7916	0.79	1.20		0.97	0.8412	0.74	1.26	
DIGOXIN_PRESCRIBED	1.47	0.0072	1.11	1.94	*	0.75	0.0845	0.56	0.99	
DISCHARGED_DISP_30_DAYS	5.54	0.0000	3.93	7.82	*	0.95	0.8588	0.59	1.53	
DISCHARGED_DISP_31_365_DAYS	4.17	0.0000	3.52	4.94	*	0.00	0.0000	0.00	0.00	
DISCHARGED_DISP_365_DAYS	4.32	0.0000	3.67	5.09	*	1.12	0.4286	0.88	1.42	
Divorced : Married	1.08	0.6221	0.80	1.46		0.00	0.0000	0.00	0.00	
EJFR_IND	3.83	0.0000	3.27	4.48	*	1.56	0.0000	1.32	1.84	*
EJFR_NUM	1.02	0.0000	1.02	1.03	*	0.00	0.0000	0.00	0.00	
ER_VISITS_365_DAYS_COPD_IND	10.14	0.0000	7.07	14.53	*	3.60	0.0000	2.48	5.23	*
FEMALE_IND	1.06	0.4825	0.90	1.24		0.99	0.9511	0.86	1.15	
HF_IND	0.98	0.8453	0.81	1.19		0.51	0.0001	0.39	0.68	*

Continued on next page

Table 3.18 – *Continued from previous page*

Variable	Univariate analysis					Multivariate analysis				
	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary				Lower boundary	Upper boundary	
HOSP_12M_VISIT	1.54	0.0000	1.49	1.59	*	1.10	0.0155	1.03	1.18	*
HOSP_30D_VISIT	3.93	0.0000	3.35	4.60	*	1.15	0.3746	0.88	1.51	
HOSP_365_DAYS_-COPD_IND	12.49	0.0000	10.42	14.96	*	3.72	0.0000	3.02	4.57	*
HTN_IND	0.91	0.2184	0.78	1.06		0.69	0.0002	0.59	0.81	*
INSULIN_PRESCRIBED	1.70	0.0001	1.31	2.21	*	0.90	0.5348	0.67	1.20	
INTUB_GT_OR_E_-2DAYS_LST_2YRS	2.75	0.0101	1.27	5.95	*	0.90	0.8125	0.43	1.87	
LOOPDIURETIC_-PRESCRIBED	1.50	0.0003	1.20	1.88	*	0.00	0.0000	0.00	0.00	
LOS_GTE_10_DAYS_-HOSP	3.41	0.0000	2.73	4.25	*	1.12	0.4377	0.88	1.42	
METOLAZONE_-PRESCRIBED	1.46	0.1391	0.88	2.43		0.92	0.7793	0.57	1.49	
MG_PCP_18M_SEEN_-IND	2.13	0.0000	1.82	2.49	*	0.00	0.0000	0.00	0.00	
MI_IND	0.91	0.6526	0.62	1.35		1.05	0.8382	0.73	1.51	
MILDOBDOP_-PRESCRIBED	1.80	0.6844	0.11	30.77		0.00	0.9665	0.00	99999	

Continued on next page

Table 3.18 – *Continued from previous page*

Variable	Univariate analysis					Multivariate analysis				
	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary				Lower boundary	Upper boundary	
NONINSULIN_DIAB_PRESCRIBED	1.36	0.0078	1.08	1.70	*	0.89	0.4886	0.67	1.18	
NOT_PRN_MED_TOTAL_PRESCRIBED	1.13	0.0000	1.11	1.15	*	0.00	0.0000	0.00	0.00	
NUM_ER_VISITS_30_DAYS_COPD	10.59	0.0000	4.60	24.39	*	1.16	0.8264	0.39	3.44	
NUM_ER_VISITS_30_DAYS_PNEU	6.32	0.0880	1.07	37.40		2.57	0.4745	0.29	22.64	
NUM_ER_VISITS_31_365_DAYS_COPD	3.76	0.0000	2.81	5.03	*	0.00	0.0000	0.00	0.00	
NUM_ER_VISITS_31_365_DAYS_PNEU	4.46	0.0023	1.99	10.02	*	0.00	0.0000	0.00	0.00	
NUM_ER_VISITS_365_DAYS_COPD	3.93	0.0000	2.98	5.18	*	0.00	0.0000	0.00	0.00	
NUM_ER_VISITS_365_DAYS_PNEU	4.71	0.0005	2.25	9.86	*	1.05	0.9433	0.37	2.95	
NUM_HF_HOSP_365_31_DAYS_COUNT	1.86	0.0000	1.59	2.17	*	0.95	0.7047	0.76	1.19	
NUM_HOSP_30_DAYS_COPD	16.43	0.0000	12.10	22.32	*	2.08	0.0028	1.39	3.11	*

Continued on next page

Table 3.18 – *Continued from previous page*

Variable	Univariate analysis					Multivariate analysis				
	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary				Lower boundary	Upper boundary	
NUM_HOSP_30_DAYS_-PNEU	6.63	0.0000	4.87	9.01	*	1.26	0.3790	0.82	1.93	
NUM_HOSP_31_365_-DAYS_COPD	4.12	0.0000	3.67	4.63	*	0.00	0.0000	0.00	0.00	
NUM_HOSP_31_365_-DAYS_PNEU	2.64	0.0000	2.40	2.90	*	0.00	0.0000	0.00	0.00	
NUM_HOSP_365_DAYS_-COPD	4.39	0.0000	3.94	4.90	*	0.00	0.0000	0.00	0.00	
NUM_HOSP_365_DAYS_-PNEU	2.61	0.0000	2.39	2.85	*	1.17	0.0630	1.02	1.35	
NUM_MISSED_APPTS_-365	1.07	0.0000	1.06	1.08	*	0.99	0.1796	0.97	1.00	
NUM_UNIQUE_-SPECIALTIES	1.39	0.0000	1.34	1.43	*	1.25	0.0000	1.19	1.30	*
O2_IND	4.49	0.0000	3.49	5.79	*	1.64	0.0014	1.27	2.11	*
ORALNITRATE_-PRESCRIBED	1.41	0.0151	1.07	1.86	*	0.74	0.0821	0.56	0.98	
ORALVASODILSPERF_-PRESCRIBED	3.21	0.0000	2.24	4.59	*	1.28	0.2981	0.87	1.89	
Other : Caucasian	1.43	0.0115	1.08	1.88	*	0.00	0.0000	0.00	0.00	

Continued on next page

Table 3.18 – *Continued from previous page*

Variable	Univariate analysis					Multivariate analysis				
	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary				Lower boundary	Upper boundary	
PAT_AGE_YRS	1.00	0.3006	1.00	1.01		1.02	0.0000	1.01	1.03	*
PL_ADHD_IND	2.48	0.1289	0.77	7.98		2.14	0.2349	0.75	6.14	
PL_BIPOLAR_IND	0.44	0.1578	0.14	1.38		0.49	0.2527	0.18	1.36	
PL_CKD_IND	1.49	0.0184	1.07	2.09	*	1.00	0.9945	0.72	1.38	
PL_DEPR_IND	0.96	0.7239	0.74	1.23		0.87	0.3276	0.69	1.10	
PL_HEPCIRR_IND	0.76	0.4370	0.37	1.53		0.54	0.1247	0.28	1.04	
PL_HIV_IND	3.16	0.1183	0.75	13.41		2.81	0.2131	0.72	11.03	
PL_PD_IND	0.82	0.6420	0.37	1.86		0.67	0.3585	0.32	1.38	
PL_PM_IND	1.42	0.0336	1.03	1.96	*	0.98	0.9006	0.70	1.35	
PL_PSYCH_IND	1.32	0.5100	0.58	2.98		1.78	0.1877	0.87	3.65	
PL_SCIATICA_IND	1.14	0.1956	0.94	1.38		0.80	0.0379	0.66	0.95	*
PL_SICKLE_IND	9.48	0.0444	1.06	84.94	*	22.53	0.0178	2.59	19575	*
PLAVIX_PRESCRIBED	1.90	0.0000	1.50	2.40	*	1.44	0.0126	1.13	1.83	*
PULMO_24MM_VISIT	1.38	0.0000	1.33	1.42	*	1.17	0.0000	1.12	1.21	*
Single : Married	1.32	0.0157	1.05	1.66	*	0.00	0.0000	0.00	0.00	
SMOKER_IND	3.19	0.0000	2.44	4.17	*	2.21	0.0000	1.74	2.80	*
SPIRON_PRESCRIBED	2.05	0.0000	1.60	2.64	*	1.21	0.2251	0.93	1.58	
STATIN_PRESCRIBED	1.50	0.0000	1.28	1.76	*	0.79	0.0198	0.66	0.93	*

Continued on next page

Table 3.18 – *Continued from previous page*

Variable	Univariate analysis					Multivariate analysis				
	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity	Odds Ratio	$Pr(\|z\| > Z)$	Confidence interval		Validity
			Lower boundary	Upper boundary				Lower boundary	Upper boundary	
THIAZIDEDIUR_- PRESCRIBED	1.72	0.0000	1.43	2.08	*	1.11	0.3772	0.91	1.35	
TOTAL_LOS_HOSP_- DAYS_LST_12_MO	1.06	0.0000	1.05	1.07	*	0.00	0.0000	0.00	0.00	
TOTAL_MEDS_- PRESCRIBED	1.12	0.0000	1.10	1.14	*	1.09	0.0000	1.06	1.12	*
Unknown : Married	0.43	0.0640	0.18	1.05		0.00	0.0000	0.00	0.00	
Widowed : Married	1.25	0.0150	1.04	1.50	*	0.00	0.0000	0.00	0.00	

As can be seen from Table 3.18, the most significant predictive variables with respect to their odd ratios are NUM_HOSP_30_DAYS_COPD, NUM_ER_VISITS_30_DAYS_PNEU, and NUM_ER_VISITS_30_DAYS_COPD. We can also observe that PAT_AGE_YRS appears to be insignificant from the point of view of the corresponding odds ratio. In spite of this, we need to bear in mind that, as pointed out in Section 3.7.2, a one year increase in patient age does not change the odds of hospitalization significantly and thus patient's age cannot be automatically discarded from the final model.

3.7.4 Assessing model coefficients

The coefficients of a linear or logistic regression are computed using a variant of the normal equation (3.10). In reality, this relationship includes the random error component

$$y = \sum_{i=0}^{N} a_i x_i + \epsilon \, , \tag{3.54}$$

$$x_0 = 1 \, , \tag{3.55}$$

where the intercept has been incorporated into the general equation for convenience by virtue of (3.54). Coefficients a_i, obtained with the help of (3.54) - (3.55), are estimates, albeit *unbiased* [21]; the uncertainty in their calculation is implied by the random nature of ϵ. If we assume the normality of errors, $\epsilon \sim \mathcal{N}(0, \sigma^2)$, then the standard null hypotheses $H_0(a_i) : a_i = 0$ can then be tested by computing the t-statistic

$$t_i = \frac{\hat{a}_i - a_{i0}}{s.e.(\hat{a}_i)} \, , \; i = \overline{1, N} \, , \tag{3.56}$$

$$s.e.(\hat{a}_i) = \sqrt{\frac{MS_{Res}}{S_{xx}}} \, , \tag{3.57}$$

$$MS_{Res} = \frac{1}{N-2} \sum_{i-1}^{N} \epsilon_i^2 \, , \tag{3.58}$$

$$S_{xx} = \sum_{i=1}^{N} (x_i - \overline{x})^2 \tag{3.59}$$

$$\overline{x} = \frac{1}{N} \sum_{i=1}^{N} x_i \tag{3.60}$$

$$t_0 = \frac{\hat{a}_0 - a_{00}}{s.e.(\hat{a}_0)} \, , \tag{3.61}$$

$$s.e.(\hat{a_0}) \quad = \quad \sqrt{MS_{Res}\left(\frac{1}{N}+\frac{\overline{x}^2}{S_{xx}}\right)}\,, \tag{3.62}$$

$$t_i \quad \sim \quad \chi^2_{N-2}\,. \tag{3.63}$$

The significance of the coefficient, i.e., the probability that it comes from a distribution centered at 0 is determined by the test statistic t_i. In view of 3.63, we can compute the appropriate p-values at the α significance level and construct the usual confidence intervals for $a_i\ , i=\overline{1,N}$ as

$$a_i \quad \in \quad \left[\hat{a_i} - t_{\frac{\alpha}{2},N-2}\times s.e.(a_i), \hat{a_i} + t_{\frac{\alpha}{2},N-2}\times s.e.(a_i)\right]\ , \tag{3.64}$$

In our ongoing COPD example, we can now finalize the set of predictive variables and create a model for testing and validation. Drawing upon the results presented in Table 3.18 and Section 3.7.1, we select the coefficients for model (3.18) based on the statistical significance of their odds ratios and subject matter knowledge, and calculate their statistics presented in Table 3.19.

Table 3.19: . Example: COPD logistic regression model coefficients and their statistics

Variable	Estimate	Std. error	Z value	$Pr(\|z\| > Z)$	Validity
(Intercept)	−677	0.37	−182	0.0000	*
ACE_INHIBITOR_-PRESCRIBED	0.20	0.10	1.93	0.0530	
ANTICOAG_-PRESCRIBED	0.21	0.11	1.97	0.0485	*
ARB_PRESCRIBED	0.18	0.12	1.58	0.1137	
ASPIRIN_PRESCRIBED	−013	0.11	−124	0.2135	
BETA_BLOCKER_-PRESCRIBED	−010	0.10	−101	0.3127	
CALCCHANBLOCKER_-PRESCRIBED	0.21	0.10	2.22	0.0266	*
DIGOXIN_PRESCRIBED	−028	0.16	−169	0.0902	
EJFR_IND	0.55	0.10	5.57	0.0000	*
ER_VISITS_365_DAYS_-COPD_IND	1.05	0.22	4.76	0.0000	*

Continued on next page

Table 3.19 – *Continued from previous page*

Variable	Esti-mate	Std. error	Z value	$Pr(\|z\| > Z)$	Va-li-di-ty
HOSP_12M_VISIT	0.12	0.03	4.24	0.0000	*
HOSP_365_DAYS_COPD_-IND	1.66	0.11	14.85	0.0000	*
INSULIN_PRESCRIBED	−014	0.17	−086	0.3903	
INTUB_GT_OR_E_-2DAYS_LST_2YRS	0.01	0.42	0.03	0.9771	
NONINSULIN_DIAB_-PRESCRIBED	−004	0.14	−032	0.7524	
NUM_UNIQUE_-SPECIALTIES	0.24	0.02	9.70	0.0000	*
O2_IND	0.56	0.15	3.77	0.0002	*
ORALNITRATE_-PRESCRIBED	−030	0.16	−184	0.0660	
PAT_AGE_YRS	0.01	0.00	3.73	0.0002	*
PL_PM_IND	−002	0.18	−009	0.9248	
PLAVIX_PRESCRIBED	0.26	0.14	1.87	0.0608	
PULMO_24MM_VISIT	0.14	0.02	6.35	0.0000	*
SMOKER_IND	0.85	0.15	5.69	0.0000	*
SPIRON_PRESCRIBED	−006	0.16	−036	0.7206	
STATIN_PRESCRIBED	−024	0.10	−243	0.0151	*

As follows from Table 3.19, HOSP_365_DAYS_COPD_IND, ER_VISITS_-365_DAYS_COPD_IND, SMOKER_IND, O2_IND and EJFR_IND have the most impact on the estimated probability of outcome of interest and are statistically significantly different from 0. From the clinical perspective, this makes prefect sense. On the other hand, automatically removing from the model those variables that are not statistically significantly different from 0 may result in loss of information and is not generally recommended.

3.8 Transformation of variables

In many instances, variable transformation does not change the qualitative nature of the relationship between the corresponding predictive variable and the outcome. In obvious cases, however, it may significantly improve the quality of the model as illustrated by the following, admittedly contrived, example.

The data in Table 3.20 was generated as $y = x^4 + \epsilon$, where $\epsilon \sim \mathcal{N}(0, 1)$. Constructing a straightforward linear regression model $y = \beta_0 + x\beta_1$ (*cf.*

X	X^4	Y
1.00	1.00	0.02
2.00	16.00	22.74
3.00	81.00	21.94
4.00	256.00	980.14
5.00	625.00	806.77
6.00	1,296.00	719.72
7.00	2,401.00	3,142.90
8.00	4,096.00	5,830.59
9.00	6,561.00	8,408.08
10.00	10,000.00	8,833.27
11.00	14,641.00	24,506.23
12.00	20,736.00	23,564.77
13.00	28,561.00	23,480.56
14.00	38,416.00	19,291.35
15.00	50,625.00	67,606.97
16.00	65,536.00	64,802.90
17.00	83,521.00	83,203.28
18.00	104,976.00	128,786.91
19.00	130,321.00	154,355.33
20.00	160,000.00	179,868.25

Table 3.20: Example: Variable transformation

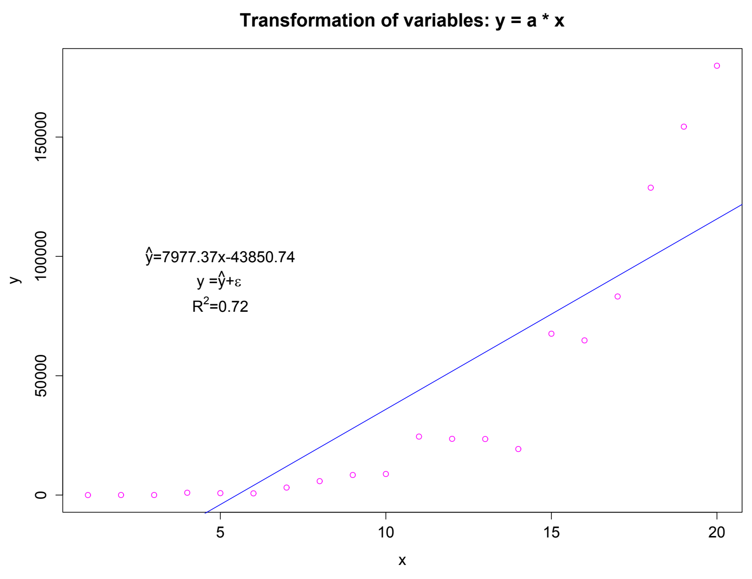
3.9) yields an expectedly poor fit depicted in Fig. 3.12a with $R^2 = 0.72$.

Performing a simple variable transformation, $\tilde{x} = x^4$ and applying a "generalized" linear model $y = \beta_0 + \beta_1\tilde{x}$ results in a much better fit with $R^2 = 0.98$, as can be seen in Fig. 3.12b.

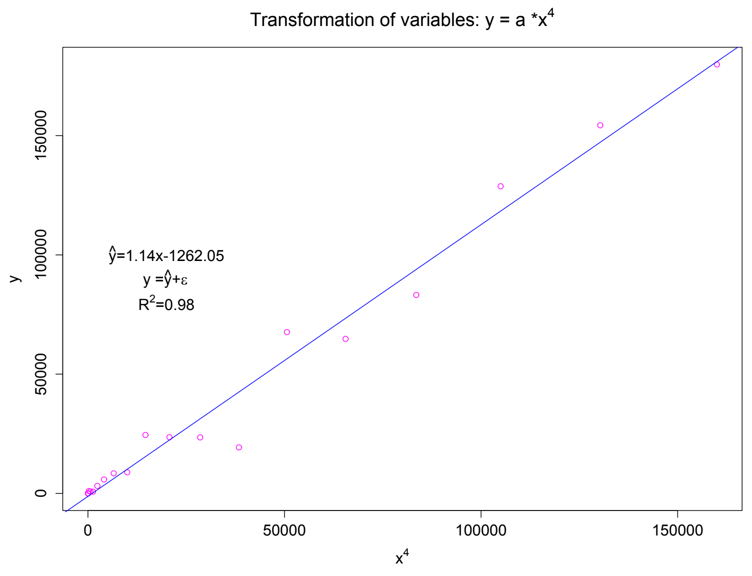
The code for generating Fig. 3.12 is presented in Listing 3.7.

3.9 Including interaction terms in the model

Common sense suggests that an optimal choice among models with approximately equal performance characteristics is the one that has the fewest "moving parts". This principal is often (simplistically) referred to as *Occam's razor*[36] and quoted as "Numquam ponenda est pluralitas sine necessitate" (Plurality must never be posited without necessity), and "Frustra fit



(a) $y = \beta_0 + x\beta_1$



(b) $y = \beta_0 + x^4\beta_1$

Figure 3.12: Example: variable transformation from x to x^4.

```
linMod <- function( x, y, fn, main, xlab, tx,
    ty ) {
  b <- fn( x )
  lm <- lm( y ~ b )
  coef <- coef( lm )
  yHat <- x * coef[2] + coef[1]
  plot( b, y, main=main, col='magenta',
     xlab=xlab )
  abline( coef=coef, col='blue' )
  a <- sprintf( "%.2f", coef[2] )
  b <- sprintf( "%.2f", coef[1] )
  snb <- ifelse( sign( coef[1] ) == 1, "+",
     "" )
  text( tx, ty[1], bquote( paste( hat( y ),
     "=", .(a), "x", .(snb), .(b) ) ) )
  text( tx, ty[2], bquote( paste( "y =", hat(
     y ), "+", epsilon ) ) )
  R2 <- sprintf( "%.2f",
     summary(lm)$r.squared )
  text( tx, ty[3], bquote( paste( R^2, "=",
     .(R2) ) ) )
}

set.seed(1)
x <- 1:20
xR <- rnorm( 1:20 )
y <- (x + xR)^4
mt <- "Transformation of variables: y = a *"
ty <- c( 1e5, 0.9e5, 0.8e5 )
linMod( x, y, I, paste( mt,  "x" ), "x", 5,
   ty )
linMod( x, y, function(x) {x^4}, bquote(
   paste( .(mt),  x^4) ), bquote( x^4 ),
    2e4, ty )
df <- data.frame( x=x, x.4=x^4, y=sprintf(
   "%6.2f", y ) )
write.csv( df, "./VarTran.csv")
```

Listing 3.7: Example: R code for linear regression.

per plura quod potest fieri per pauciora" (It is futile to do with more what can be done with less)[32]. In agreement with this principle, we generally prefer linear models to their nonlinear counterparts as long as their performance metrics do not differ significantly. There are cases, however, when a linear model simply will not do (see, e.g., the example in Section 3.8). We are not aware of any universal recipe for selecting a specific variable transformation in every possible instance. If there are sufficient reasons to suspect from general subject domain considerations that predictive variables may influence each other, introducing interaction terms may improve model performance.

Table 3.21 illustrates a contrived example of hospitalization data for a hypothetical population of patients. For each patient in Table 3.21, both age

ID	Age	Sex	Hospitalization	ID	Age	Sex	Hospitalization
1	84	M	1	16	63	M	1
2	69	F	1	17	74	M	1
3	74	M	1	18	78	F	1
4	69	F	1	19	69	F	1
5	56	F	0	20	79	F	1
6	66	M	1	21	74	F	1
7	66	F	1	22	64	F	0
8	69	F	1	23	73	F	1
9	81	F	1	24	59	M	1
10	78	M	1	25	84	F	1
11	68	F	1	26	90	M	1
12	67	F	1	27	66	M	1
13	77	M	1	28	60	M	1
14	76	M	1	29	76	M	1
15	63	F	0	30	69	F	1

Table 3.21: Example: Hypothetical hospitalizations.

and sex were generated randomly, and only females whose age is at or above the median age of the sample less 5 were hospitalized[1]. Since the data is random by construction, we are at liberty to use the first 20 rows of the table for training and the remaining 10 rows for testing our models. The results

[1]The median age of the sample is 69, therefore all females over 64 have $hospitalization = 1$.

of applying a strictly linear model of the form *hospitalization* $\sim$ *age* + *sex* to the testing dataset are presented in Fig. 3.13. The results of applying a

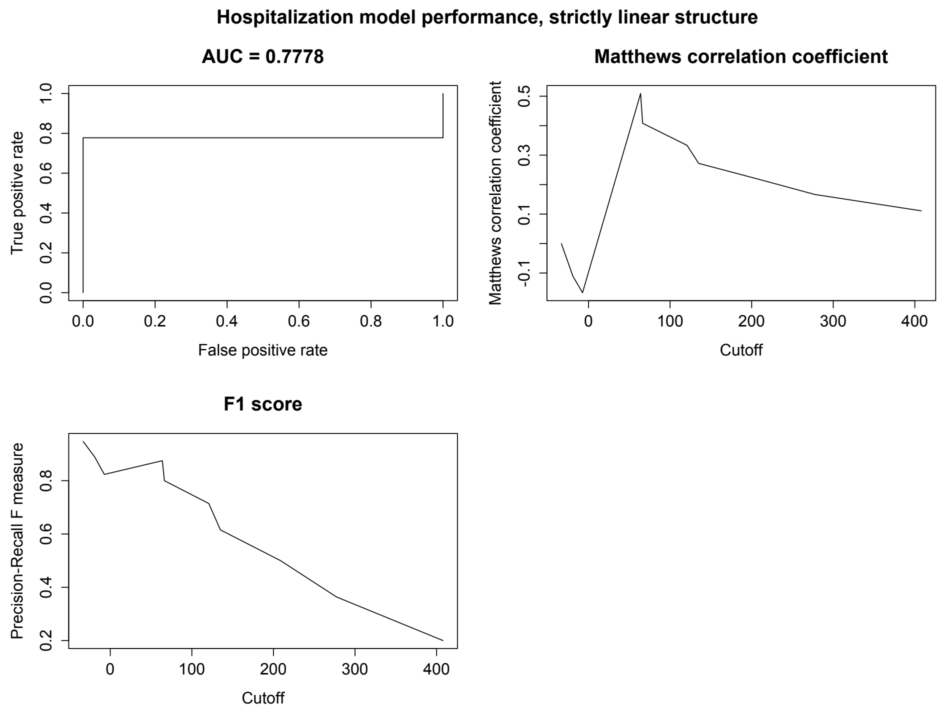


Figure 3.13: AUC, F_1 score score and Matthews' correlation coefficient of the strictly linear logistic regression model for the hypothetical hospitalization example in Table 3.21.

linear model with an interaction term of the form *hospitalization* $\sim$ *age* + *sex* + *age* $\times$ *sex* to the testing dataset are presented in Fig. 3.14. Not surprisingly, the performance of the model without the interaction terms (ROC curve AUC of 0.78 in Fig 3.13) is inferior to that of the model with interaction terms included (ROC curve AUC of 1.00 in Fig 3.14), and the use of the more complicated model is justified.

The code for generating Figs. 3.13 and 3.14 is presented in Listing 3.8.

3.10 Model validation

Once a model has been developed, it has to be validated to ensure that it meets development specifications. Regardless of the type of the model, it has

```
linMod <- function( train, test, inclCol, outCol, sep,
    main ) {
  form <- formula( paste( outCol, "~", paste( inclCol,
      collapse=sep ) ) )
  gm <- glm( form, data=train, family='binomial' )
  prediction <- predict( gm, test )

  perf <- auc.perf.base( prediction, test[, outCol],
     text=main )
}

set.seed(1)
n <- 30
id <- 1:n
gender <- rnorm( id )
age <- round( 70 + 10 * rnorm( id ) )
sex <- ifelse( gender <= 0, "M", "F" )
y <- ifelse( ( age - median( age ) ) * ifelse( sex ==
   'M', 0, 1 ) <= -5, 0, 1 )
mt <- "Interaction term: age * gender"
data <- data.frame( ID=id, age=age, sex=sex,
   hospitalization=y )
trainRows <- 1:round( n * 2 / 3 )
testRows <- ( max( trainRows ) + 1 ):n
test <- data[testRows, ]
train <- data[trainRows, ]

main <- "Hospitalization model performance, strictly
   linear structure"
linMod(train, test, c( "age", "sex" ), "
   hospitalization", '+', main )
main <- "Hospitalization model performance,
   interaction terms included"
linMod(train, test, c( "age", "sex" ), "
   hospitalization", '*', main )

write.csv( data, "./IntTermEx.csv" )
```
Listing 3.8: Example: Hypothetical hospitalizations.

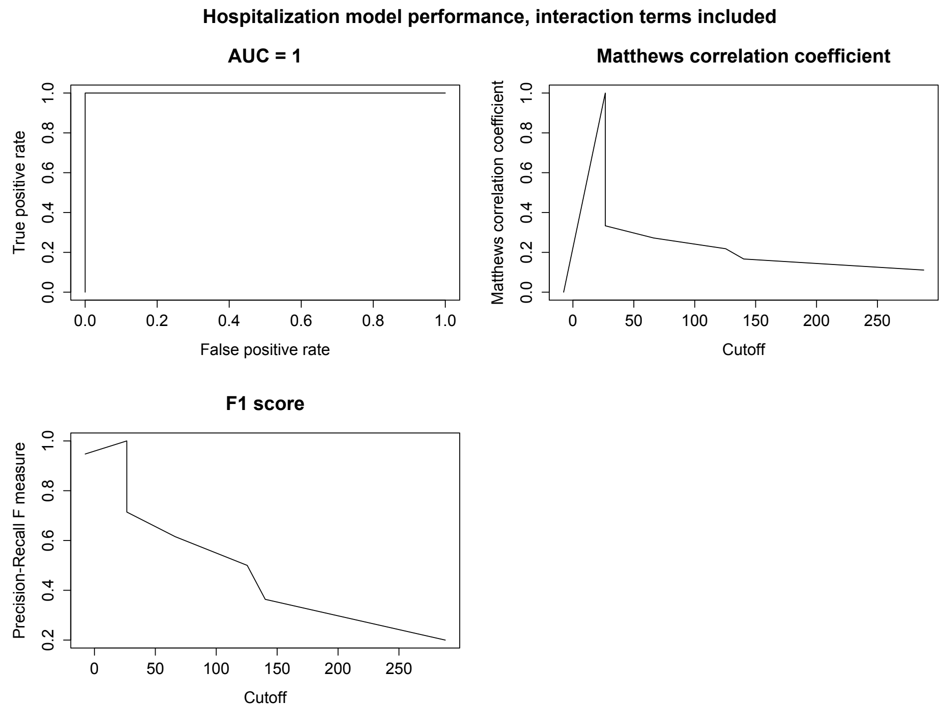


Figure 3.14: AUC, F_1 score score and Matthews' correlation coefficient of the logistic regression model with a cross term for the hypothetical hospitalization example in Table 3.21.

to be cross-validated on an independent data set, and the results compared with the training dataset to detect possible under- or overfitting. Specific validation methods for the types of model most commonly used by Clinical Analytics are described in the rest of this section.

3.10.1 Linear regression

Linear regression assumes the existence of a linear relationship between the input variables and the observed output. In general, a successful model must satisfy several requirements to be considered acceptable as a predictive tool [23]:

i sufficiently high R^2 (typically, at least 0.7) - this will confirm that a large proportion of variation in the dependent variable can be explained by the variation in the independent variable(s);

ii reasonably good visual fit between the straight line predicted by the model and the actual functional relationship between the dependent and independent variables;

iii sufficiently random residuals (at least no noticeable trend)

Once these requirements have been satisfied, the model can be deemed sufficiently accurate for our needs.

3.10.2 Logistic regression

Logistic regression is a classification model and thus needs to be evaluated on its ability to predict the outcome of interest. One of the most intuitive and widely accepted techniques for this purpose is computing the area under the receiver operating characteristics (ROC) curve. We adopt it as a universal measure of fit for classification models of any nature, including logistic regression. In a typical example of time-dependent data the preferred way is to proceed as follows:

a build a regression model on the selected training set (80% of all data);

b use the model to predict outcomes on the testing set (20% of all data);

c compute the area under the curve (AUC) for the corresponding ROC;

d if the AUC is acceptable, separate the dataset onto the ”old” and ”new” data (e.g., all years up to 1 year ago and the most recent year) and repeat the test;

e if AUCs from different datasets are comparable and the differences between them can be reasonably explained, accept the model, otherwise, go back to the drawing board and repeat.

If the model allows backward transition from the outcome of interest (admissions), the training and test datasets can be generated from patient data using multiple observations of the same patient; in the opposite case (mortality), a random data point is chosen from the patient's time-dependent data. This approach can be justified by observing that if a patient can experience the outcome of interest multiple times, each encounter can be viewed as an independent event with a possible outcome of interest. If a patient can only experience the outcome of interest once, the use of the same patient's data accumulated over the years violates the assumption of independence [11] between observations[1] and, additionally, ascribes disproportionally high weight to those who did not experience such outcome thus leading to potential "survivor bias" [13][2].

In order to provide confidence interval boundaries for AUC to facilitate the comparison of model quality, an appropriate estimation technique needs to be selected. The most accurate estimates are based on the parametric assumption of binormality for the AUC curve [37]. Such an assumption is not unduly restrictive for large datasets, and the obtained estimates employ the usual z-statistic argument. When the number of positive outcomes is relatively small[3], a semi-parametric or nonparametric estimate may be desirable. For our purposes, we deem it sufficient to construct the confidence interval for the AUCs by using repeated sampling as described in step 2 of the algorithm in Section 3.10.3.

Other measures of goodness-of-fit include (see Section 3.12 below) *F_1 score* and *Matthews' correlation coefficient*. These are presented as supplementary metrics for the purpose of identifying the optimal balance between true positive rate and specificity and usually complement each other.

Continuing with the example in Section 3.7.1, Fig. 3.15[4] presents the AUC, F_1 score and Matthews' correlation coefficient for the logistic regression model for predicting hospitalizations in COPD patients previously referenced in Table 3.19[5]. As can be seen from Fig. 3.15, the AUC for the model in question is approximately 0.75. The optimal balance between sensitivity and specificity is attained at the cutoff point of approximately 10% of the population. In other words, it appears optimal to flag approximately 1/10th of the patients as being at high risk of admission for COPD-related reasons and, if the objective is efficient case management, concentrate limited resources allocated to this task on this subgroup.

It is considered good practice to compare the results of a developed model with a benchmark "null hypothesis" option whenever possible. For example, if the object of our investigation is assessment of relative hospital-

[1]Clearly, if a patient's clinical data can be observed in year 3, it implies that he or she was alive in years 1 and 2.

[2]An alternative opinion [30] states that conditional probability of survival embedded in "person-periods" allows for their treatment as if they were independent. Out of abundance of caution, we chose not to adopt this argument. Using the Cox proportional hazard model or *generalized estimating equations* [19] (GEE) eliminates such controversy.

[3]30 or fewer for each type of outcome of interest for moderate AUCs and 150 or fewer for $AUC \geq 0.95$

[4]Fig. 3.15 was generated using `run.glm.model` presented in Appendix E.4 and illustrated by Appendix C.

[5]Model performance metrics were calculated on the test dataset.

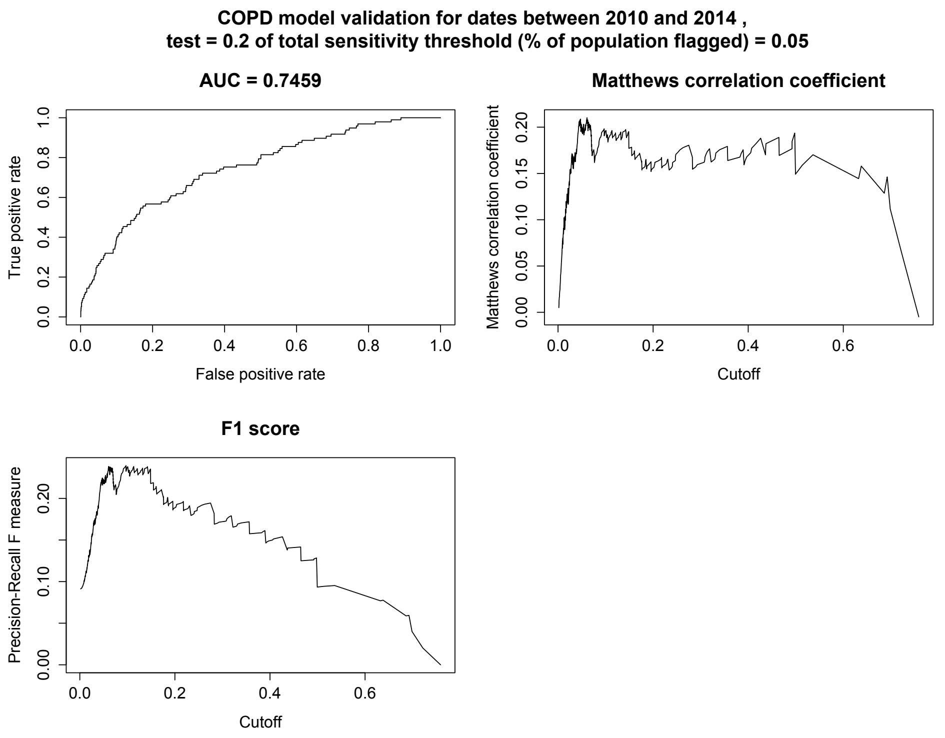


Figure 3.15: AUC, F_1 score score and Matthews' correlation coefficient of the logistic regression model for COPD.

ization risk for a group of patients, the corresponding benchmark could be random selection from the total population of a sample equal in size to our group. Concretely, suppose that we have developed such a model based on the Elixhauser approach [10] and selected a "naïve random" benchmark as described above. Table 3.22 summarizes the results of applying each model to the total patient population and selecting 1,000 with the highest risk score[1]. The superiority of the Elixhauser model is evident: 24.4% of the

Model	Hospitalizations related to CHF, CVA, COPD, CAD or DM Foot, % of total	Observational or Inpatient Hospitalizations (Non-Pregnancy / Trauma), % of total	*AUC*
Random guess	0.80	8.20	0.50
Logistic regression	24.40	57.60	0.93

Table 3.22: Example: Comparison of the Elixhauser admission model with the random selection benchmark for 1,000 riskiest patients.

1,000 patients most likely to be admitted were actually admitted to the hospital during the subsequent year for specified diagnoses, compared to only 0.8% of those selected randomly. For general admissions those figures are 8.2% and 57.6% respectively. AUC comparison yields 0.5 for the random guess (as expected) and 0.93 for the Elixhauser model (excellent).

3.10.3 Validation of temporal datasets[2]

When working with patient data, it is common to consider time-dependent outcomes for the same individual as separate dataset entries unless an outcome of interest presents an absorbing boundary, i.e., is irreversible (e.g., in mortality risk modeling[3]). It is thus pertinent to ask what set of tests is sufficient to convince a reasonably skeptical examiner[4] that a newly devel-

[1] i.e., "probability of admission"; this "probability" should be understood in a relative sense.

[2] In our effort to keep up with contemporary technical literature, we favor the spelling of *dataset* over *data set* (cf. *database* vs. now obsolete *data base*).

[3] We are considering mortality purely from the point of view of modern medicine.

[4] see, e.g., [24]

oped model works universally well under practical circumstances. While the answer to this question is often subjective, the following testing routine has so far yielded satisfactory results for the purpose of identifying intervention candidates in the total population health management program:

1. separate the final usable dataset into the training and testing portions by designating a random 20% sample for testing and the remaining 80% for training the model;

2. repeatedly run the final model (as described by in Fig. 3.7) on training datasets obtained at the previous step until satisfied with the goodness-of-fit statistics;

3. execute the *forward* test as

   (a) train the model on the first available period[1] of data;

   (b) from the remaining period data, select the entries that are appropriate under the assumption that no posterior information is available and no patient data is given disproportional weight in the model, e.g., by selecting only one period data point of patient data at random in the mortality model;

   (c) refine the original model as necessary until AUCs for testing period data are satisfactory;

4. execute the *backward* test as

   (a) train the model on the last available period of data;

   (b) from the remaining period data, select the entries in the same way as for the forward test;

   (c) refine the original model as necessary until AUCs for testing period data are satisfactory;

5. execute the *mid history* test as

   (a) train the model on the first and last available periods of data;

   (b) from the remaining period data, select the entries in the same way as for the forward test;

   (c) refine the original model as necessary until AUCs for testing period data are satisfactory;

[1]most often, year

6. execute the *last available period* test as

    (a) from all periods except the last one for which full outcome of interest data is available, select the entries in the same way as for the forward test;

    (b) train the model on the data selected in the preceding step;

    (c) test the model on the last period for which full outcome of interest data is available;

    (d) refine the original model as necessary until AUCs for testing period data are satisfactory;

3.11 Predicting future outcomes

Once the test program outlined in Section 3.10.3 has yielded consistent AUC estimates and reasonably stable coefficients[1], the "production", or "forward-looking", model is constructed by training the algorithm on the whole dataset[2].

Sample code for an implementation of the prediction algorithm is given in Appendices C ("vanilla" logistic regression), D (Cox proportional hazard model) and E.4. The algorithms apply the respective vectors of regression coefficients (3.19) or (3.29) to generate the appropriate risk score ("probability" of outcome of interest for linear regression or hazard function for the Cox proportional hazard model). Once a risk rating has been assigned to every member of the test sample, they can be ranked by their ratings in descending order. N riskiest members can then be selected from the population as candidates for intervention.

Table 3.23 presents 10 patients at highest risk of COPD admission from the test population of 2,049 in the ongoing example from Section 3.7.1.

In Table 3.23 "Risk" is the probability of outcome of interest given by the logistic regression model.

3.12 Evaluating model performance

The Clinical Analytics team uses receiver operating characteristic (ROC) curve as the main metric for evaluating the performance of a logistic regression or Cox proportional hazard model. For evaluating the optimal balance

[1]Consistency and stability here are determined from business and "common sense" considerations rather than mathematical estimates.

[2]i.e., by expanding the training dataset in the first step of the algorithm to 100% of the data and shrinking the testing dataset to nothing.

Patient	Risk
1	0.98
2	0.97
3	0.97
4	0.96
5	0.95
6	0.94
7	0.94
8	0.93
9	0.92
10	0.92

Table 3.23: Example: 10 patients at highest risk of COPD admission from a population of 2,049.

between true positive rate and specificity, F_1 score and Matthews' correlation coefficient are also included.

3.12.1 Receiver operating characteristics curve

ROC curves are an essential tool for assessing the quality of a classification model. Table 3.24 illustrates the relationship between the actual and predicted outcomes as reflected by such curves.

		Predicted outcome		
		p	**n**	**total**
actual outcome	**p′**	true positive rate	false negative	P′
	n′	false positive	specificity	N′
	total	P	N	

Table 3.24: Tabularized relations between truth/falseness of the null hypothesis and outcomes of the test

ROC curve graphs are used for internal research purposes and for presentation to technical audiences familiar with this concept. The ROC curve for a survival model describing the mortality risk in heart failure patients is presented as an example in Fig. 3.16.

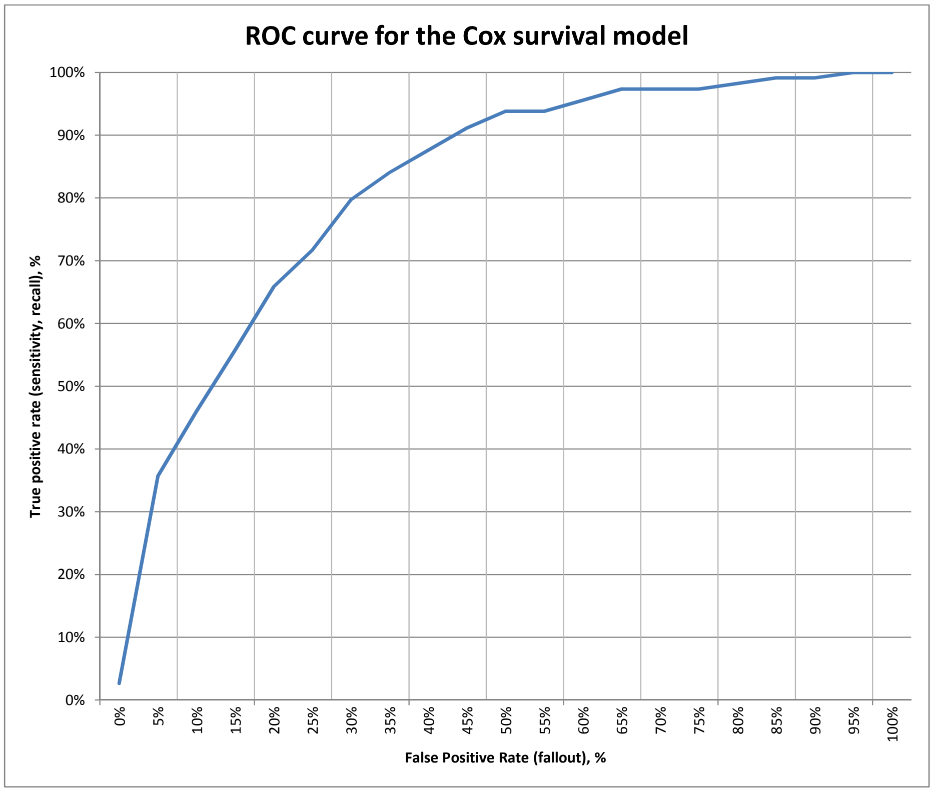


Figure 3.16: Sample receiver operating characteristic (ROC) curve graph.

An example of a plot combining ROC, F_1 score and Matthews' correlation coefficient was presented earlier in Fig. 3.6

For historical reasons, the Clinical Analytics team has found it more instructive and easily digestible for executive and practitioner audiences to employ combined lift curve / positive predictive value graphs as a tool for visualizing the quality of a predictive model. lift curve / positive predictive value graphs are presented double-scaled with true positive rate plotted on the left in red, and positive predictive value on the right in blue. The abscissa (x-axis) represents the percentage of the (test) population that was classified by the model as having an outcome of interest. The combined graph for a logistic heart failure admission prediction model is presented as an example in Fig. 3.17.

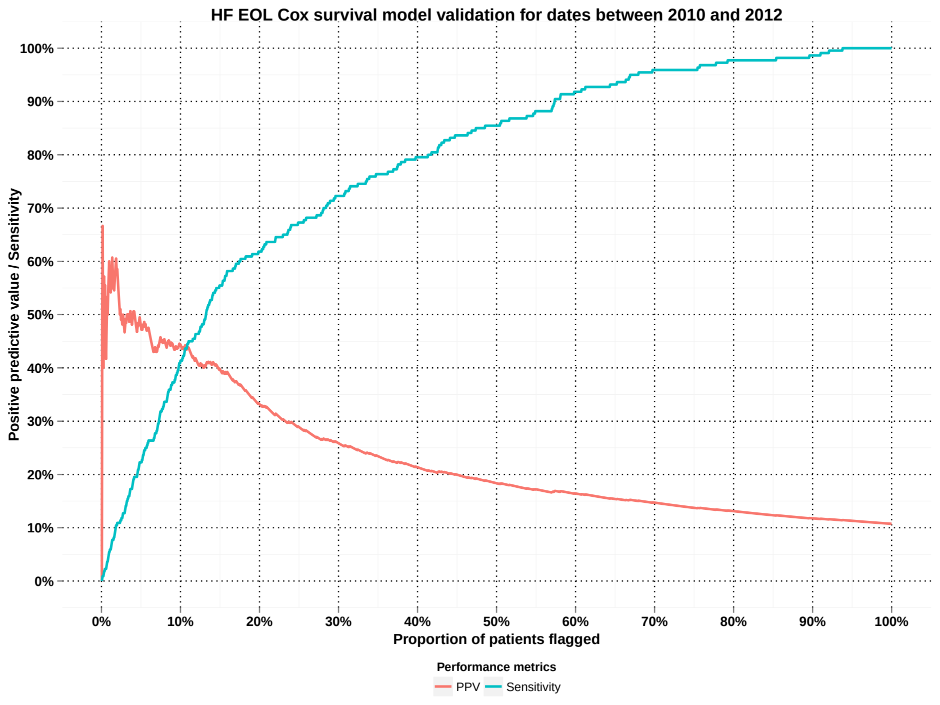


Figure 3.17: Sample lift curve graph.

As follows from Fig. 3.17, should the riskiest 5% of the patients selected by the model be chosen for intervention, true positive rate in that population will be approximately 50%. This rate reaches approximately 10% in the general population, hence the lift achieved by applying the model for the top 5% is close to 5.

3.12.2 Summary of model performance metrics

In order to evaluate model performance, it is helpful to summarize some of the relevant model metrics in one place. Table 3.25 is an extension of Table 3.23 that includes the corresponding performance parameters calculated for the same 10 patients on the original test dataset.

In Table 3.23 columns have the following meanings:

Patient	Risk	Actual outcome	# flagged	% flagged	PPV	specificity	lift
1	0.98	1	1	0.05%	1.00	0.00	9.31
2	0.97	0	2	0.10%	0.50	0.00	4.66
3	0.97	0	3	0.15%	0.33	0.00	3.10
4	0.96	0	4	0.20%	0.25	0.00	2.33
5	0.95	1	5	0.24%	0.40	0.01	3.73
6	0.94	1	6	0.29%	0.50	0.01	4.66
7	0.94	1	7	0.34%	0.57	0.02	5.32
8	0.93	1	8	0.39%	0.63	0.02	5.82
9	0.92	0	9	0.44%	0.56	0.02	5.17
10	0.92	0	10	0.49%	0.50	0.02	4.66

Table 3.25: Example: 10 patients at highest risk of COPD admission from a population of 2,049 with model performance metrics included.

Risk	-	probability of outcome of interest given by the logistic regression model;
Actual outcome	-	actual observed outcome;
# flagged	-	number of patients on the given and preceding rows for whom the model predicted positive outcomes;
% flagged	-	percentage of such patients relative to the total population (2,049 individuals);
PPV	=	positive predictive value.

The output template featured in Table 3.25 is adopted by the Clinical Analytics team as the preferred way of illustrating the model performance. This layout is easy to present and explain to the upper management in order to facilitate operational business decisions, including those concerning resource allocation.

If the model with an irreversible outcome described in Section B is tested on a randomly sampled dataset that includes a single entry for surviving patients, an argument can be made that the positive predictive value of the model in Table 3.25 is inflated by underweighting survivor's data. This concern can be addressed by selecting the data from the most recent time period (e.g., year) as the test dataset[1] and using it to calculate the corresponding model performance metrics in this table. If this approach is followed, all

[1]i.e., using all data, excluding the most recent period, with one randomly selected entry per each multiple-year survivor as the training dataset.

data in the test population will be reduced to a single entry per patient with equal weights. Linear regression models described in Section 3.2 and Cox proportional hazard models described in Section 3.4 do not require this adjustment.

4 Presentation of results

4.1 Output data storage

The storage model for output data should facilitate the achievement of the following objectives:

- keep project-related data together in a form that is
  - compact,
  - logical,
  - readable and
  - easily accessible;
- make it easy to perform unit testing;
- help compare the results of incremental changes in the code;
- support creating time snapshots of the model for auditing purposes.

The above-mentioned objectives can be accomplished more easily if

1. readable output files are stored in the `Results` directory (as mentioned in Section 6.2),
2. graphs (PDFs, JPGs etc.) are stored in the `Graphs` subdirectory of `Results`
3. both readable files and graphs are cataloged by date of the corresponding program run in separate directories named `yyyy-mm-dd` with appropriate commentary appended separated by underscores (e.g., `2014-05-14_Hip_Knee`)

4.2 Output data naming conventions

All output files are named using abbreviated functional descriptions of their contents and are date stamped for subsequent reference. Naming conventions for output files are listed in Table 4.1.

Output type	File type	Naming convention	Example
Model coefficients and their statistics	.csv	Coeff_All_[yyyy_mm_dd].csv	Coef_All_2014_10_28.csv
Predicted risk probabilities and model performance statistics for all patients of interest	.csv	AllRisk_[yyyy_mm_dd].csv	AllRisk_2014_10_28.csv
Predicted risk probabilities and model performance statistics for high risk patients	.csv	HiRisk_[yyyy_mm_dd].csv	HiRisk_2014_10_28.csv
Predicted risk probabilities and model performance statistics for 100 highest risk patients	.csv	HiRiskRand100_[yyyy_mm_dd].csv	HiRiskRand100_2014_10_28.csv
PPV, specificity and F_1 score for 100 highest risk patients	.csv	HiRiskRand100_[yyyy_mm_dd].csv	HiRiskRand100_2014_10_28.csv
PPV and sensitivity lift curves	.pdf	PPV_Sense_Full_[yyyy_mm_dd].pdf	PPV_Sens_Full_2014_10_28.pdf
ROC, F_1 score and Matthews' correlation coefficient curves	.pdf	AUC_[yyyy_mm_dd].pdf	AUC_2014_12_03.pdf

Table 4.1: Naming convention for output files.

4.3 Presentation format

Most of the research projects carried out by the Department of Clinical Analytics produce data outcome that can best be digested by the audience if presented in the form of tables and graphs. While it is difficult to prescribe a universal format for a successful table, it is nevertheless desirable to establish the broadest possible documentation standards some of which are listed below.

I Output files

  i numeric output that will be ingested into Excel or R for further processing should be stored as `.csv` files,
  ii plain text files should be avoided whenever possible,
  iii if extended markup is desired (e.g., web browser output), `XML` output is appropriate,
  iv where extensive data post-processing manipulation is anticipated, a (sandbox) database table for the results is desirable;

II Graphs

  i preferably, graphs should be stored as `PDF` files with axes, legend and tick marks clearly labeled and easily readable (in general, 12 pts. or larger),
  ii landscape orientation is preferred,
  iii legend coloring scheme should be consistent with that of the plot itself.

Documentation not requiring extensive mathematical formulae, sophisticated graphics or cross-referencing can be created in Microsoft Word. Papers that do require substantial typesetting should be created in LaTeX, if possible.

5 Data storage

5.1 Input data storage

Interim input data can be stored as text, CSV or XML files, Excel spreadsheets or sandbox databases. For consistency, it is preferable to keep input data in the "Data" folder of the corresponding project folder. When the data is intended for use by other people, it is helpful to use a single format that can be easily picked up and converted into a form convenient for its consumer. For most practical purposes, CSV is preferred. If the data is stored in a sandbox database, SQL scripts used for data extraction can be stored in the "Code" folder of the project. If the creation of a shared internal database is desirable and possible for the purpose of the project, it can be set up on the common server with tables named for specific tasks.

5.1.1 Nomenclature of input variables

In an effort to standardize the nomenclature of input variables, a suggested list of common names is presented in Table 5.1.

Table 5.1: Input variable nomenclature

Predictive variable	Type	Meaning
ACE_INHIBITOR_PRESCRIBED	indicator	active angiotensin-converting-enzyme inhibitor prescription
ADN_EVER	indicator	active advanced directive note
AICD_IND	indicator	automated implantable cardioverter-defibrillator
ANTICOAG_PRESCRIBED	indicator	active anticoagulant prescription
ARB_PRESCRIBED	indicator	active angiotensin receptor blocker prescription
AS_OF_DATE	date	date of record
ASPIRIN_PLAVIX_PRESCRIBED	indicator	active aspirin and clopidogrel prescription indicator
ASPIRIN_PRESCRIBED	indicator	active aspirin prescription
BETA_BLOCKER_PRESCRIBED	indicator	active beta-blocker prescription
BMI__06_MO	continuous	body mass index, 6 months ago
BMI__12_MO	continuous	body mass index, 12 months ago
BMI__24_MO	continuous	body mass index, 24 months ago
BMI__36_MO	continuous	body mass index, 36 months ago
BMI__48_MO	continuous	body mass index, 48 months ago
BMI__MR	continuous	body mass index, most recent
BNP__06_MO	continuous	brain natriuretic peptide, 6 months ago
BNP__12_MO	continuous	brain natriuretic peptide, 12 months ago
BNP__24_MO	continuous	brain natriuretic peptide, 24 months ago
BNP__36_MO	continuous	brain natriuretic peptide, 36 months ago

Continued on next page

Table 5.1 – *Continued from previous page*

Predictive variable	Type	Meaning
BNP__48_MO	continuous	brain natriuretic peptide, 48 months ago
BNP__MR	continuous	brain natriuretic peptide, most recent
CAD_IND	indicator	cardioarterial disease
CALCCHANBLOCKER_-PRESCRIBED	indicator	active calcium channel blocker prescription
CANCER_DX	indicator	cancer diagnosis
CANCER_IND	indicator	cancer
CARD_MOST_SEEN_IND	indicator	most seen by a cardiologist
CARDO_24MM_VISIT	continuous	# of times seen by a cardiologist in the last 24 months
CARDO_12MM_VISIT	continuous	# of times seen by a cardiologist in the last 12 months
CHOL__06_MO	continuous	total cholesterol level, 6 months ago
CHOL__12_MO	continuous	total cholesterol level, 12 months ago
CHOL__24_MO	continuous	total cholesterol level, 24 months ago
CHOL__36_MO	continuous	total cholesterol level, 36 months ago
CHOL__48_MO	continuous	total cholesterol level, 48 months ago
CHOL__MR	continuous	total cholesterol level, most recent
COMBOANTHYP_PRESCRIBED	indicator	active combined antihypertensive prescription indicator
COPD_IND	indicator	chronic obstructive pulmonary disorder
CREAT__06_MO	continuous	creatinine, 6 months ago
CREAT__12_MO	continuous	creatinine, 12 months ago
CREAT__24_MO	continuous	creatinine, 24 months ago
CREAT__36_MO	continuous	creatinine, 36 months ago
CREAT__48_MO	continuous	creatinine, 48 months ago

Continued on next page

Table 5.1 – *Continued from previous page*

Predictive variable	Type	Meaning
CREAT__MR	continuous	creatinine, most recent
CSO_EVER	continuous	do-not-resuscitate standing order at any time
DAYS_SINCE_LAST_EF	continuous	# of days since the last ejection fraction test was ordered
DBP__06_MO	continuous	diastolic blood pressure, 6 months ago
DBP__12_MO	continuous	diastolic blood pressure, 12 months ago
DBP__24_MO	continuous	diastolic blood pressure, 24 months ago
DBP__36_MO	continuous	diastolic blood pressure, 36 months ago
DBP__48_MO	continuous	diastolic blood pressure, 48 months ago
DBP__MR	continuous	diastolic blood pressure, most recent
DEATH_365_DAYS	indicator	death occurred within the 365 days following AS_OF_DATE
DEATH_365D_IND	indicator	death occurred within the 365 days following AS_OF_DATE
DEATH_AS_OF_DATE	date	date of death if occurred within the 365 days following AS_OF_DATE
DEATH_DATE	date	date of death
DEM_IND	indicator	dementia
DIAB_IND	indicator	diabetes
DIGOXIN_PRESCRIBED	indicator	active digoxin prescription
DISCHARGED_DISP_30_DAYS	continuous	# of hospital discharges in the last 30 days
DISCHARGED_DISP_31_365_DAYS	continuous	# of hospital discharges between the last 31 and 365 days
DISCHARGED_DISP_365_DAYS	continuous	# of hospital discharges in the last 365 days

Continued on next page

Table 5.1 – *Continued from previous page*

Predictive variable	Type	Meaning
EJFR_DATE	date	date of ejection fraction test
EJFR_NUM	continuous	ejection fraction value
FIRST_AICD_DATE	date	date of first implantation of an automated implantable cardioverter-defibrillator
FIRST_HF_DX_DATE	date	date of first heart failure diagnosis
FRAM_TOTAL_POINTS	continuous	Framingham risk score
GENDER_CODE	categorical	gender
HDL __06_MO	continuous	high-density lipoprotein cholesterol, 6 months ago
HDL __12_MO	continuous	high-density lipoprotein cholesterol, 12 months ago
HDL __24_MO	continuous	high-density lipoprotein cholesterol, 24 months ago
HDL __36_MO	continuous	high-density lipoprotein cholesterol, 36 months ago
HDL __48_MO	continuous	high-density lipoprotein cholesterol, 48 months ago
HDL __MR	continuous	high-density lipoprotein cholesterol, most recent
HEMOGL_MR	continuous	hemoglobin, most recent
HF_IND	indicator	heart failure
HOSP_12M_VISIT	continuous	# of hospital visits in the last 12 months
HOSP_30D_VISIT	continuous	# of hospital visits in the last 30 days
HTN_IND	indicator	hypertension
INP_OBS_COPD_ADM_365_DAYS	indicator	admitted to the hospital for observation for COPD-related issues in the last 365 days
INP_OBS_HF_365_DAYS	indicator	admitted to the hospital for HF-related issues in the last 365 days

Continued on next page

Table 5.1 – *Continued from previous page*

Predictive variable	Type	Meaning
INP_OBS_HF_ADM_365_DAYS	indicator	admitted to the hospital for observation for HF-related issues in the last 365 days
INSULIN_PRESCRIBED	indicator	active insulin prescription
INTUB_GT_OR_E_2DAYS_LST_-2YRS	indicator	two or more intubation days in the last two years indictor
IVNITRATE_PRESCRIBED	indicator	IV nitrate
IVVASODILSPERF_PRESCRIBED	indicator	IV vasodilator
LDL __06_MO	continuous	low-density lipoprotein, 6 months ago
LDL __12_MO	continuous	low-density lipoprotein, 12 months ago
LDL __24_MO	continuous	low-density lipoprotein, 24 months ago
LDL __36_MO	continuous	low-density lipoprotein, 36 months ago
LDL __48_MO	continuous	low-density lipoprotein, 48 months ago
LDL __MR	continuous	low-density lipoprotein, most recent
LOOPDIURETIC_PRESCRIBED	indicator	active loop diuretic prescription indicator
LOS_GTE_10_DAYS_HOSP	indicator	hospitalization with length of stay exceeding 10 days indicator
MARITAL_STATUS	categorical	marital status
METOLAZONE_PRESCRIBED	indicator	active metolazone prescription
MG_ONC_VISIT	indicator	visited a medical group oncologist
MG_PCP_18M_SEEN_IND	indicator	medical group physician visited within the last 18 months
MG_PCP_24M_SEEN_IND	indicator	medical group physician visited within the last 24 months
MI_IND	indicator	myocardic infarction

Continued on next page

Table 5.1 – *Continued from previous page*

Predictive variable	Type	Meaning
MILDOBDOP_PRESCRIBED	indicator	active mild organic brain disease opioid prescription indicator
MR_CARD_DATE	date	most recent cardiologist visit
MR_CONTACT_DATE	date	most recent contact date
MR_PACEMAKER_DATE	date	most recent Pacemaker implantation date
MR_PHY_ID	ID	ID of the most recent attending physician
MR_VISIT_MG_IND	indicator	attended to by a medical group physician during the most recent hospital visit
MR_VISIT_PRACTICE	categorical	practice name of the physician during the most recent hospital visit
MR_VISIT_PRIM_SPEC	categorical	primary specialty of the physician during the most recent hospital visit
NONINSULIN_DIAB_PRESCRIBED	indicator	active non-insulin diabetes prescription
NOT_PRN_MED_TOTAL_-PRESCRIBED	continuous	active non-pro-re-nata medication prescription
NUM_CARD_SEEN_MOST_LST_-24_MO	continuous	# of times seen by a cardiologist in the last 2 years
NUM_ER_VISITS_30_DAYS_COPD	continuous	# of Emergency Department visits for COPD in the last 30 days
NUM_ER_VISITS_30_DAYS_PNEU	continuous	# of Emergency Department visits for pneumonia in the last 30 days
NUM_ER_VISITS_31_365_DAYS_-COPD	continuous	# of Emergency Department visits for COPD between the last 31 and 365 days

Continued on next page

Table 5.1 – *Continued from previous page*

Predictive variable	Type	Meaning
NUM_ER_VISITS_31_365_DAYS_-PNEU	continuous	# of Emergency Department visits for pneumonia between the last 31 and 365 days
NUM_ER_VISITS_365_DAYS_COPD	continuous	# of Emergency Department visits for COPD in the last 365 days
NUM_ER_VISITS_365_DAYS_PNEU	continuous	# of Emergency Department visits for pneumonia in the last 365 days
NUM_HF_HOSP_365_31_DAYS	continuous	# of hospital visits for heart failure between the last 31 and 365 days
NUM_HOSP_30_DAYS	continuous	# of days spent in the hospital in the last 30 days
NUM_HOSP_30_DAYS_CHF	continuous	# of days spent in the hospital for congestive heart failure in the last 30 days
NUM_HOSP_30_DAYS_COPD	continuous	# of days spent in the hospital for COPD in the last 30 days
NUM_HOSP_30_DAYS_PNEU	continuous	# of days spent in the hospital for pneumonia in the last 30 days
NUM_HOSP_31_365_DAYS_COPD	continuous	# of days spent in the hospital for COPD between the last 31 and 365 days
NUM_HOSP_31_365_DAYS_PNEU	continuous	# of days spent in the hospital for pneumonia between the last 31 and 365 days
NUM_HOSP_365 _DAYS_CHF	continuous	# of days spent in the hospital for congestive heart failure between in the last 30

Continued on next page

Table 5.1 – *Continued from previous page*

Predictive variable	Type	Meaning
NUM_HOSP_365_DAYS	continuous	# of days spent in the hospital between in the last 30
NUM_HOSP_365_DAYS_COPD	continuous	# of days spent in the hospital for COPD in the last 365 days
NUM_HOSP_365_DAYS_ICU	continuous	# of days spent in the intensive care unit in the last 365 days
NUM_HOSP_365_DAYS_PNEU	continuous	# of days spent in the hospital for pneumonia in the last 365 days
NUM_KCC_VISIT	continuous	# of visits to Kellogg Cancer Center
NUM_MISSED_APPTS_3_YRS	continuous	# of missed appointments in the last 3 years
NUM_MISSED_APPTS_365	continuous	# of missed appointments last year
NUM_UNIQUE_SPECIALTIES	continuous	# of unique specialties of physicians seen by the patient
O2_IND	indicator	active oxygen prescription indicator
ORALNITRATE_PRESCRIBED	indicator	active oral nitrate prescription
ORALVASODILSPERF_-PRESCRIBED	indicator	active oral vasodilator prescription indicator
PACEMAKER_IND	indicator	Pacemaker implanted
PAT_AGE_YRS	continuous	patient age
PAT_MRN_ID	ID	patient MRN ID
PCP_ID	ID	primary care physician ID
PCP_MG_IND	indicator	has a primary care physician from the medical group
PCP_PRAC_NAME	categorical	primary care physician practice name
PCP_PRIM_SPEC	categorical	primary care physician specialty

Continued on next page

Table 5.1 – *Continued from previous page*

Predictive variable	Type	Meaning
PL_ADHD_IND	indicator	ADHD
PL_BIPOLAR_IND	indicator	bipolar disorder
PL_CKD_IND	indicator	chronic kidney disease
PL_DEPR_IND	indicator	depression
PL_HEPCIRR_IND	indicator	hepatic cirrhosis
PL_HIV_IND	indicator	HIV/AIDS
PL_PD_IND	indicator	Parkinson's disease
PL_PM_IND	indicator	psychosomatic medicine patient indicator
PL_PSYCH_IND	indicator	psychosis
PL_SCIATICA_IND	indicator	sciatica
PL_SICKLE_IND	indicator	sickle cell anemia
PLAT__MR	continuous	platelets, most recent
PLAVIX_PRESCRIBED	indicator	active clopidogrel prescription
PULMO_24MM_VISIT	continuous	# of times seen by a pulmonologist in the last 24 months
RACE_ETHNICITY	categorical	race and ethnicity category
SBP__06_MO	continuous	systolic blood pressure, 6 months ago
SBP__12_MO	continuous	systolic blood pressure, 12 months ago
SBP__24_MO	continuous	systolic blood pressure, 24 months ago
SBP__36_MO	continuous	systolic blood pressure, 6 months ago
SBP__48_MO	continuous	systolic blood pressure, 48 months ago
SBP__MR	continuous	systolic blood pressure, most recent
SBP_MR	continuous	systolic blood pressure, 6 months ago
SMOKER_IND	indicator	active smoking
SODIUM__MR	continuous	sodium, most recent
SPIRON_PRESCRIBED	indicator	active spironolactone prescription indicator

Continued on next page

Table 5.1 – *Continued from previous page*

Predictive variable	Type	Meaning
STATIN_PRESCRIBED	indicator	active statin prescription indicator
STATUS	indicator	alive or dead
TEN_YR_RISK_PCT	continuous	10-year resk percentage
THIAZIDEDIUR_PRESCRIBED	indicator	active thiazide diuretic prescription indicator
TOTAL_LOS_HOSP_DAYS_LST_12_-MO	continuous	total length of hospital stays in days in the last 12 months
TOTAL_MEDS_PRESCRIBED	continuous	total # of medications prescribed
TRIGL__06_MO	continuous	triglycerides, 6 months ago
TRIGL__12_MO	continuous	triglycerides, 12 months ago
TRIGL__24_MO	continuous	triglycerides, 24 months ago
TRIGL__36_MO	continuous	triglycerides, 36 months ago
TRIGL__48_MO	continuous	triglycerides, 48 months ago
TRIGL__MR	continuous	triglycerides, most recent
VISIT_PHY_ID	ID	ID of the physician seen during the last visit
VISIT_PHY_MG_IND	indicator	physician seen during the last visit is from the medical group
VISIT_PRACTICE_NAME	categorical	practice name of the physician seen during the last visit
VISIT_PRIM_SPEC	categorical	primary specialty of the physician seen during the last visit
WEIGHT__06_MO	continuous	weight, 6 months ago
WEIGHT__12_MO	continuous	weight, 12 months ago
WEIGHT__24_MO	continuous	weight, 24 months ago
WEIGHT__36_MO	continuous	weight, 36 months ago
WEIGHT__48_MO	continuous	weight, 48 months ago
WEIGHT__MR	continuous	weight, most recent

5.2 Formatting and storage of intermediate results

During the execution of a predictive analytics script, intermediate files are stored in the project directory tree in the directory titled "Results/Interim". Intermediate plots are stored as PDF files, tables are saved as CSV files. All output files are named using abbreviate functional descriptions of their contents and are date stamped in order to preserve research history and ensure reproducibility of the results. Naming conventions for interim output files are listed in Table 5.2.

6 Coding practices

6.1 Revision control

Revision control is essential for incremental and cooperative development. Git is currently the suggested tool of choice for implementing a robust framework for sharing and improving the code. A centralized local Git repository for the Clinical Analytics team is not available at the time of this writing (February 2020), however, interim measures can be taken to ensure that changes made to the code are at least traceable to its developer(s). Collaboration Portal is currently used as a proxy for the centralized code repository, and all current code should periodically be placed on the portal to facilitate quality control and cross-training of the department team members. In preparation for the implementation of the centralized Git repository, all team members should implement Git framework on their local computers and regularly check in working code with appropriate descriptive comments. If this practice is followed, the creation of a centralized Git node will be reduced to pushing individual repositories to the designated location and should take place with minimum resource diversion from higher priority tasks.

Once the central repository is set up, one person (presumably, the project manager) should be designated as the administrator with one or two team members serving as backup resources fully cross-trained on the system functionality. The following is a suggested list of good repository maintenance practices in the form of do's:

- DO take regular snapshots of COMPILABLE CODE;
- DO write concise, informative, itemized comments for each commit highlighting the most significant changes from the previous version;
- DO minimize the time you keep the code checked out;

Output type	File type	Naming convention	Example
Univariate odds ratios for indicator variables	.csv	OR_[yyyy_mm_-dd].csv	OR_2014_10_28.csv
Univariate odds ratios for all variables	.csv	UniOR_[yyyy_-mm_dd].csv	UniOR_2014_10_-28.csv
Odds ratios for specific categorical variables	.csv	[Variable Name]_-[yyyy_mm_dd].csv	Ethnicity_2014_-10_28.csv, MaritalStatus_-2014_10_28.csv
Multivariate odds ratios for indicator variables	.csv	OR_All_[yyyy_-mm_dd].csv	OR_2014_10_28.csv
Highly correlated indicator variables displayed as a matrix	.csv	CorInd_Prior_-[yyyy_mm_dd].csv	CorInd_Prior_-2014_12_11.csv
Highly correlated indicator variables displayed as pairs	.csv	CorInd_-PriorTable_[yyyy_-mm_dd].csv	CorInd_-PriorTable_2014_-12_11.csv
Highly correlated continuous / interval variables displayed as a matrix	.csv	CorNum_Prior_-[yyyy_mm_dd].csv	CorNum_Prior_-2014_12_11.csv
Highly correlated continuous / interval variables displayed as pairs	.csv	CorNum_-PriorTable_[yyyy_-mm_dd].csv	CorNum_-PriorTable_2014_-12_11.csv
Baseline hypothesis quality data	.csv	NullHyp_[yyyy_-mm_dd].csv	NullHyp_2014_10_-28.csv
Correlation matrix level plot for indicator variables	.pdf	Corr_Ind_[yyyy_-mm_dd].pdf	Corr_Ind_2014_12_-11.pdf
Correlation matrix level plot for continuous / interval variables	.pdf	Corr_Num_[yyyy_-mm_dd].pdf	Corr_Num_2014_-12_11.pdf
Correlation matrix level plot for all variables	.pdf	Corr_Mat_All_-[yyyy_mm_dd].pdf	Corr_Mat_All_-2014_12_11.pdf

Table 5.2: Naming convention for interim output files.

- DO conduct unit tests before checking in the code to make sure it is backward compatible.
- DO merge branches at the first opportunity.

and don't:

- DON'T check in code that does not compile;
- DON'T check in code that will break the build;
- DON'T store the executable, compiled, auxiliary or any other binary files with the source;
- DON'T create more branches than necessary.

(see, e.g., [9], [4], [20]).

6.2 Code storage

The following is a suggested directory structure for storing project code:

- Methodology
  - methodology documents and white papers describing the algorithm;
  - testing and implementation procedures;
  - production implementation requirements;
- Data
  - input data organized by run date, functionality or model version as appropriate;
  - tools (e.g., Excel spreadsheets) for pre-processing input data (if applicable);
- Code
  - project files (if applicable);
  - source code differentiated by language (if applicable):
    - R
    - Python
    - SQL

        * C#
        * Other;

- Log

    - log and error files (if applicable);

- Results

    - output data files by run date, functionality or model version as appropriate;
    - graphical output by run date, functionality or model version as appropriate;

6.3 Code review

Peer review is an indispensable code verification and validation tool that also facilitates the development of robust, scalable and reusable code. Once a developer has completed a new release to his or her satisfaction, they should initiate code review with a designated peer. The assignment of peers can be very informal, especially when the new model has been confined to the domain of narrow expertise. It is considered beneficial to the quality of the algorithm to have a person less familiar with the methodology review and, time permitting, replicate the results of the newly shipped release. In the absence of a designated QA department, the only defense against inadvertent flaws in the code is the institution of a process that requires the algorithm's author to fully explain the methodology and coding decisions behind it to "skeptical" colleague. Such colleague should understand the basic concepts but not be biased in any way towards accepting the result. Resources permitting, having more than one person review the code would strengthen the quality control process - but we have to be realistic about what we can expect of ourselves given more pressing time commitments.

Once the code has been review by a designated "tester", it can be tagged as "production version" in the repository thus becoming an official release.

6.4 Naming conventions

If words of command are not clear and distinct, if orders are not thoroughly understood, the general is to blame.

attributed to Sun Tzu[34]

Names of objects used throughout the coding code should be clear, concise, consistent and descriptive. Suggested naming conventions are detailed in Table 6.1.

Table 6.1: Naming conventions

Language	Object type	Scope	Conflicts	Naming convention	Comment
R	variable	any scope	none	capitalLetterDividers	
	function	local	none	capitalLetterDividers	project-specific functions not tested for generic use
		packaged	no naming conflict with external packages	func.that.works	use simple dot separation if no external package functions with the same name exist
			same name as a function from an external package	NS.CA.func.that .works	prepend NS.CA (NorthShore Clinical Analytics) to a dot-separated function name
	package	global	none	NS.CA.capitalLetterDividers	mark packages as developed by NorthShore Clinical Analytics

Continued on next page

Table 6.1 – *Continued from previous page*

Language	Object type	Scope	Conflicts	Naming convention	Comment
	project	global	none	SentenceCase_with_underscore	abbreviations in ALLCAPS, project functionality in SentenceCase, separation by underscores
	file	local	none	runProjectName.R	highest-level function ("Dispatcher")
		packaged	no naming conflict with external packages	packageName.R	minimally descriptive short names are encouraged
			same name as an external package	NS.CA.packageName.R	prepend NS.CA to the package name

6.5 Writing quality code

The definition of what constitutes quality code in any programming language could be the subject of a lengthy debate that is best carried out away from volatile compounds and other easily inflammable materials. Below follow a few fundamental principles that the original writers of this document believe to be universal and rarely disputed.

6.5.1 R

- Write readable code:
  - create a high level function that calls analytical and auxiliary functions as needed;
  - reference packages only where the use of such packages is required at the lowest level;

  - indent your code
  - use spaces around operators, after commas, after opening and before closing braces and parentheses;
  - wrap long lines at column 80 (remember the punch cards? I'm only partially joking here...);
  - use `knitr`-style comments;
  - in a function, first list the required, then the optional parameters.

- Write meaningful comments:
  - in a function
    - explain what the function is for;
    - describe input and output arguments;
    - list any specific parameter values that present special cases.

    The mode function in the `NS.CA.statUtils` package is an example:

```
## ----- NS.CA.mode -----
## Mode(s) of the distribution
## Usage
### NS.CA.mode(x, fun=function(y) {y})
## Arguments
### x - matrix or data frame containing the
    distribution(s) (convert to matrix if list)
### fun - function determining which mode to
    select in the multimodal case

NS.CA.mode <- function(x, fun=function(y) {y}) {
  ux <- unique(x)
  t<-tabulate(match(x, ux))
  fun(ux[which(t == max(t))])
}
```

  - in a loop
    - mark nested long loops if necessary;
    - document "forks" as appropriate
  - in an `if-then-else` structure
    - explain what the logic means when necessary;

        * mark matching braces as needed.

- Favor `sapply`, `lapply` and `ddply` over `for`;
- Above all, DO NOT copy and paste! If a piece of code is used more than once, turn it into a function instead.
- Avoid `rbind` wherever possible since it can be slow.
- `Reduce` early and often, e.g.,

```
# aggregated charges and costs

  totalChgAll[[aggregator]] <- Reduce( function(...)
     merge(..., by=totalGroupBy, all=T, suffixes=
     totSuff), totChgCostsNotNull)
```

Accepted naming conventions are listed in Table 6.1.

6.6 Testing and QA

The design stage of application development is an excellent time to ensure that subsequent testing and validation of the model progress as smoothly as possible. Many of the issues arising at a later stage can be mitigated by ensuring open communication channels between development and production teams by remembering that an ounce of prevention is worth a pound of cure:

- have a list of candidate predictor variables for the research project;
- find out which variables are available from the historical dataset - and which are not;
    - if a variable has been consistently available throughout history, find out whether its meaning has changed;
    - if a variable has appeared only recently, find out how it can be synthesized from past data. **Ensure that the way you are replicating the new variable from the data warehouse is consistent with the way it is currently being generated. It will save you a lot of time and headaches.**
- ensure that the test data set can be easily replicated by the data warehouse team;

  - if an easy, one-to-one mapping between your dataset and theirs is hard to achieve, change your dataset, if possible;
  - if your dataset must be constructed in a specific way, get the data warehouse team started on matching your data extract as early as possible.

An application can be productionalized efficiently not only through writing correct, clean and efficient code but also through carrying out as many testing and data reconciliation iterations as possible within a limited time frame. This can be achieved by following the general guidelines below:

1. develop unit test framework whenever possible;
2. fix and freeze the input data for reconciliation testing to ensure reproducible test results;
   (a) hard-code seeds for the code dependent on random number generators;
   (b) take a snapshot of the input data at a point in the past and use it for subsequent calculations at least until the current round of testing is finished;
3. rank all differences between the old and new results in descending order and
4. drill down into the "worst offenders" until a satisfactory explanation of the differences can be found and an acceptable level of accuracy can be achieved.

Appendix A Additional tools for data analysis

A.1 Comparing two datasets

Like a home inspector in the world of real estate, a data scientist employed in the area of care standardization has no friends among physicians. In order to avoid being hit from behind by a baseball bat on their way to the car after a long day at the office, he or she must ensure that potentially damning conclusions they reached during the course of the said long day are statistically sound. Here is how:

i identify potential outliers;

ii attempt to explain and clean out spurious outliers, e.g., convert dates formed using two-digit years to proper $YYYY$ dates;

iii remove remaining outliers if you must or incorporate them into your dataset;

iv backfill missing data;

v identify and prune datasets that are not statistically significantly different from each other, e.g., physicians' pharmacy charges where the hypothesis about the two providers charging substantially identical amounts cannot be rejected at the 5% level.

There are several tests for determining whether the difference between two or more data sets can be viewed as statistically significant [35]. A summary ([17], parts reproduced by permission from Professor James D. Leeper) is presented in Table A.1.

Table A.1: Statistical tests used to confirm the statistical significance of difference between data sets; parts reproduced from [17] by permission from Professor James D. Leeper

Number of dependent variables	Number of independent variables	Independent variable type	Dependent variable type	Test	R code example
1	0 (one population)	any	normal (continuous)	one-sample t-test	`t.test(x, ...)`
			ordinal variable or interval variable	one-sample median	`median(x, ...)`
			indicator variable	binomial test	`binom.test(n, ntr, p, alt="gr")`
			categorical variable	chi-square goodness-of-fit	`chisq.test(x,...)`
	1 (independent groups)	indicator variable	normal (continuous)	two independent sample t-test	`t.test(x, y,...)`
			ordinal variable or interval variable	Wilcoxon-Mann-Whitney test	`wilcox.test(x, y,...)`
			categorical variable	Chi-square test	`chisq.test(x,...)`
				Fisher's exact test	`fisher.test(x,...)`
	1 (independent groups)	categorical variable	normal (continuous)	one-way ANOVA	`aov(x, y,...)`
			ordinal variable or interval variable	Kruskal-Wallis test	`kruskal.test((x, y,...)`

Continued on next page

Table A.1 – *Continued from previous page*

Number of dependent variables	Number of independent variables	Independent variable type	Dependent variable type	Test	R code example
1			categorical variable	chi-square test	`chisq.test(x,...)`
	1 (dependent / matched groups)	interval variable	normal (continuous)	paired t-test	`t.test(x, y, paired=T)`
			ordinal variable or interval variable	Wilcoxon signed ranks test	`wilcox.test(x, y, paired=T)`
			categorical variable	McNemar test	`mcnemar.test(x,...)`
	1 (dependent / matched groups)	categorical variable	normal (continuous)	one-way repeated measure ANOVA	`car::Anova(model, ...)`
			ordinal variable or interval variable	Friedman test	`friedman.test(y, groups, blocks,...)`
			categorical variable	repeated measures logistic regression	`glmer(x,...)`
	2+ (independent groups)	categorical variable	normal (continuous)	factorial ANOVA	`aov(model,...)`
			ordinal variable or interval variable	ordered logistic regression	`MASS::polr(model, ...)`
			categorical variable	factorial logistic regression	`glm(x,...)`
1	1	continuous	normal (continuous)	correlation	`cor(x,y, method="pearson",...)`
				simple linear regression	`lm(model,...)`

Continued on next page

Table A.1 – *Continued from previous page*

Number of dependent variables	Number of independent variables	Independent variable type	Dependent variable type	Test	R code example
			ordinal variable or interval variable	non-parametric correlation: Spearman's ρ or Kendall's τ	`cor(x,y method="spearman", ...)` or `cor(x,y, method="kendall", ...)`
			categorical variable	simple logistic regression	`glm(x,...)`
	1+ coninuous and/or 1+ categorical variable	continuous and/or categorical variable	normal (continuous)	multiple regression	`lm(model,...)`
				analysis of covariance	`aov(model,...)`
			categorical variable	multiple regression	`lm(model,...)`
				discriminant analysis	`MASS:lda(model, ...)` or `MASS:qda(model, ...)`
2+	1	categorical variable	normal (continuous)	one-way MANOVA	`manova(model,...)`
	2+	any	normal (continuous)	multivariate multiple linear regression	`cor(x,...)`
	0	any	normal (continuous)	factor analysis	`cor(x,...)`
2+ sets of 2+	0	any	normal (continuous)	Pearson correlation	`cor(x,...)`

A.1.1 Testing proportions

A question often arises in the course of comparing results of medical treatments, "Are the differences in proportions of outcomes of interest observed in different populations due to chance?" A variation on the same theme is, "Is the difference between the hypothesized and observed proportions of outcomes of interest due to chance?"

In the most general case, sources (e.g., [22]) recommend using the **pooled two-proportion z-test** for ascertaining that the difference in percentages of positive outcomes between two samples is *not* due to chance. It proceeds as follows:

i formulate the null hypothesis: $H_0 : p_1 = p_2$, where $p_i, i = 1, 2$ is the i-th proportion of outcomes of interest;

ii calculate proportions of outcomes of interest in each sample: $\hat{p_1}$ and $\hat{p_2}$;

iii calculate the total combined (pooled) proportion of outcomes of interest: $\hat{p} = \frac{n_{p_1} + n_{p_2}}{n_1 + n_2}$, where $n_{p_i}, i = 1, 2$ is the number of positive outcomes in the i-th sample, $n_i, i = 1, 2$ is the number of observations in the i-th sample;

iv calculate the standard error of the estimated difference $\hat{p_1} - \hat{p_2}$:
$SE = \sqrt{\hat{p}\,(1 - \hat{p})\left(\frac{1}{n_1} + \frac{1}{n_2}\right)}$;

v calculate the z-statistic: $z = \frac{\hat{p_1} - \hat{p_2}}{SE}$

vi calculate the $p-$value for the z-statistic;

vii if the p-value is lower than the chosen threshold (e.g., 5%), reject H_0, i.e., assume that the samples come from different distributions, otherwise accept it, i.e., assume that the samples came from the same distribution.

This test is generally applicable when the samples are no bigger than 10% of the total population and the numbers of successes and failures exceed 5.

Arguments can be made in favor of using an unpooled statistic since the variances of two samples do not have to be the same (see, e.g., [31]). In this case, the null hypothesis is $H_{0_{unpooled}} : p_1 - p_2 = d_0$, where d_0 is hypothesized difference between the two distributions, and the standard error estimate is
$SE_{unpooled} = \sqrt{\frac{\hat{p_1}(1-\hat{p_1})}{n_1} + \frac{\hat{p_2}(1-\hat{p_2})}{n_2}}$,

and the corresponding z-statistic is $\frac{\hat{p_1}-\hat{p_2}-d_0}{SE_{unpooled}}$. While there are finer points in arguing for the use of the unpooled test, for most practical cases the samples are assumed to come from the same distribution and the use of the pooled test is justified.

Let us consider a worked example based on the prevalence of blood transfusions in two hospitals (pavilions)[1] for the two-year period between May 1, 2012 and May 1, 2014 as presented in Table A.2.

Pavilion	# of patients	# of transfusions
Pavilion A	5,000	250
Pavilion B	3,000	200

Table A.2: Sample blood transfusion statistics for Q2 FY2014.

Applying our (pooled) algorithm, we get:

$$\begin{aligned}
n_1 &= 5,000 \ , \\
n_2 &= 3,000 \ , \\
n_{p_1} &= 700 \ , \\
n_{p_2} &= 200 \ , \\
\hat{p_1} &= 0.14 \ , \\
\hat{p_2} &= 0.0667 \ , \\
\hat{p_1} - \hat{p_2} &= 0.0733 \ , \\
\hat{p} &= \frac{700+200}{5,000+3,000} = 0.1125 \ , \\
SE &= \sqrt{0.1125(1-0.1125)\left(\frac{1}{700}+\frac{1}{200}\right)} = 0.028 \ , \\
z &= \frac{0.0733}{0.028} = 2.64 \\
p - value &= 2\Phi(z) = 2 \times 0.0042 = 0.0084 \ ,
\end{aligned}$$

where $\Phi(z)$ is the standard normal cumulative distribution function and the coefficient 2 comes from the two-tailed test.

Given the data in the example and the assumptions made in Section 3, the null hypothesis H_0 that the difference in proportions of patients receiving blood transfusions between pavilions A and B are highly statistically

[1] We will be using the terms "pavilion" and "hospital" interchangeably throughout this document.

significant (at higher than 99% significance level). We must therefore reject the null hypothesis H_0 and adopt the alternative H_A , i.e., assume that those proportions come from different distributions.

Appendix B Sequence of steps for developing a logistic regression model in R

Table B.1 outlines the steps to develop a logistic regression model for predicting outcomes of interest or classifying objects and events.

Table B.1: An algorithm for developing a logistic regression-based prediction or classification model

Action	Section reference	Code example	Result
Retrieve data from an outside source (database, file or website) into a data frame	App. C, D	COPDadmRaw <− read.csv(paste(dataDir, ”COPD_ALL_ALIVE.csv”, sep=””)) con <− odbcConnect(”Oracle”, uid=”user”, pwd=”passwd”) elixTrainRaw <− sqlQuery(con, ”SELECT * FROM DATATABLE”)	raw input data
Identify output variable(s)	App. C, D	outcomes <− c(”DEATH_365_DAYS”)	output variable column names in the main data frame
Convert NA to 0 where appropriate	3.5	adm[, NAzCol] <− sapply(adm[, NAzCol], function(x) ifelse(is.na(x), 0, x))	imputed sparse indicator variables
Identify indicator columns	App. C, D	indCol <− grep(”TOTAL”, grep(”_IND|_PRESCRIBED|_GT”, names(adm), value=T), invert=T, value=T)	indicator variable column names in the main data frame
Separate the data into the training and testing dataset (create cross-validation sets if necessary)	3.2	testRows <− createDataPartition(adm[, statusCol], p=testFrac, list=F)	training and testing data frames
Remove columns with sparse data (>20% missing from the training dataset)	3.2	dataSet <− rem.sparse.col(adm, trainRows, rcntCol, 0.2)	training and testing datasets without sparse variables

Continued on next page

Table B.1 – *Continued from previous page*

Action	Section reference	Code reference	Result
Remove single-valued (in the training dataset) indicator columns	App. C, D	dataTrain <– rem.single.val.col(dataSet[trainRows,])	training and testing datasets without uninformative variables
Aggregate categorical variables as needed	3.7.2	ethnicitySummary <– multi.category.stats(dataTrain, "RACE_ETHNICITY", outcomes, sensThreshold, "Caucasian", "Other", c("0", "1"))	categorical predictive variables aggregated into groups with smoothed population distribution
Compute univariate odds ratios	3.7.2	uniOddsRatio <– odds.ratio.save(dataTrain, outcomes, indCol, idCol, addList=list(eTs, mSt), oRdir=interimDir)	univariate odds ratios and their statistics for all predictive variables; intermediate files for indicator, numeric and combined indicator and numeric predictive variables saved separately
Compute multivariate odds ratios	3.7.3	coefOut <– odds.ratio.stat(linearModel, uniOddsRatio, sigLev)	multivariate odds ratios and their statistics for all predictive variables; intermediate files for predictive variables saved

Continued on next page

Table B.1 – *Continued from previous page*

Action	Section reference	Code reference	Result
Remove predictive variables whose odds ratio are statistically significantly different from 1	3.3	indSignif <- na.omit(uniOddsRatio[uniOddsRatio$Validity == '*', "Variable"])	predictive variables whose odds ratios are likely to differ from 1
Compute correlation matrices	3.3	hiCorTab <- output.corr(dataTrain, indCol, dir=interimDir, timeStamp=timeStamp, corrPlotDir=graphDir)	correlation matrices for predictive variables; intermediate files and correlation level plots saved
Remove highly correlated predictive variables ($r \geq 0.6$)	3.7.1	exclCol <- c("COMPLICATED_HYPERTENSION", "UNCOMPLICATED_DIABETES")	numerical stable normal equation (3.10)
Restore predictive variables that should be included from the business standpoint	3.3	inclCol <- c(outcomes[outInd], indSignif[indSignif %in% colnames(dataSet)])	correlation matrices for predictive variables; intermediate files and correlation level plots saved
Train the generalized lineal model (GLM) on the training dataset and run it on the test dataset	3.2, 3.3 3.7.4	fullModel <- glm.test(dataFact, statusCol, testRows, threshold, trace=trace, maxit=maxit, text=testText, varJoiner=varJoiner, plotFile=plotTestFile, ...)	logistic regression model created; model coefficients and their statistics saved
Generate ROC curves	3.12.1	res <- auc.perf(glmFit, testLm, testStatus, type=type, ...)	ROC curve plots generated and saved

Continued on next page

Table B.1 – *Continued from previous page*

Action	Section reference	Code reference	Result
Generate PPV / sensitivity curves	3.12.1	allRisk <– ppv.sens.risk(dataSet, prSort, scenario, baseText, graphDir=graphDir, resultDir=resultDir, timeStamp=timeStamp)	PPV / Sensitivity curve plots generated and saved
Generate lift tables	3.11, 3.12.1	lift.table.save(prSort, nrow(dataSet), rep(today, nrow(elix)), today, allRisk, c(0.01, 1e3 / length(testRows), 0.05, 0.1, 0.2, 0.3, 0.5, 1), resultDir=resultDir, timeStamp=timeStamp)	Lift tables saved
If performance metrics satisfy initial model specifications, submit the model for further validation and productionalization	3.10, 3.10.3		final model available for production

Appendix C Sample code used to process the running COPD example

In the listing below, the formatting is different from that of the actual code due to a different line length in the typeset font. User-defined packages texttttNS.CA.statUtils, NS.CA.dataUtils, NS.CA.dateUtils, NS.CA.modelUtils, NS.CA.plotUtils and `NS.CA.mathUtils` are defined in Appendix E.

```
### Daniel Chertok
### (C) 2014 NorthShore University HealthSystem
### All rights reserved

rm(list=ls())

library(debug)
library(reshape2)
library(date)
library(stringr)
library(plyr)
library(glmnet)
library(MASS)
library(scales)
library(lubridate)

library(NS.CA.statUtils)
library(NS.CA.dataUtils)
library(NS.CA.dateUtils)
library(NS.CA.modelUtils)
library(NS.CA.plotUtils)
library(NS.CA.mathUtils)

SPARSE_MIN <- 0.2

yearCsvName <- function( dataDir, name, year ) {
  paste(dataDir, name, year, ".csv", sep="")
}

# main module
```

```
years <- 2010:2014
today <- as.Date("2014-01-01")
dataDir <- "../Data/"
resultDir <- "../Results/"
graphDir <- paste( resultDir , "Graphs" , sep="" )
interimDir <- paste( resultDir , "Interim/" , sep="" )
timeStamp <- timestamp.from.date()

# read the input file

COPDadmRaw <- read.csv( paste(dataDir , "COPD_ALL_ALIVE
    .csv" , sep="") )

outcomes <- c(
  "INP_OBS_COPD_ADM_365_DAYS" ,
  "DEATH_365D_IND"
)

# all dates will be used later

keyDates <- sort(unique(as.Date(COPDadmRaw[ , "AS_OF_
    DATE"])))

# for analysis , use data up to today only

COPDadmRaw <- transform( COPDadmRaw, AS_OF_DATE=as.
    Date(AS_OF_DATE) )
COPDadm <- subset( COPDadmRaw, AS_OF_DATE < today )

# columns irrelevant to the analysis

excludeCol <- c("PAT_ID" , "TEN_YR_RISK_PCT" , outcomes
    [[2]])

# columns where NAs can be turne into 0s

NAzero <- c("IND" , "PRESCRIBED" , "VISIT" , "DAYS" , "NUM
    ")
NAzeroCol <- unique( Reduce( c, sapply( NAzero, grep,
    colnames(COPDadm) ) ) )
```

```
COPDadm[, NAzeroCol] <- sapply(COPDadm[, NAzeroCol],
    function(x) ifelse(is.na(x), 0, x) )

# independent of train / test breakdown

COPDadm <-
    transform( COPDadm,
               EJFR_IND=ifelse( EJFR_NUM > 0, 1, 0 ),
               FEMALE_IND=
                   ifelse( GENDER_CODE=="F", 1, 0 ),
               HOSP_365_DAYS_COPD_IND=
                   ifelse( NUM_HOSP_365_DAYS_COPD > 0,
                       1, 0 ),
               ER_VISITS_365_DAYS_COPD_IND=
                   ifelse( NUM_ER_VISITS_365_DAYS_COPD >
                       0, 1, 0 ) )
COPDadm[is.na(COPDadm$MARITAL_STATUS),"MARITAL_STATUS"
    ] <- "Unknown"
COPDadm[is.na(COPDadm$RACE_ETHNICITY),"RACE_ETHNICITY"
    ] <- NS.CA.mode(COPDadm$RACE_ETHNICITY)
ftpPts <- na.to.colMeans(data.frame(ftp=COPDadm$FRAM_
    TOTAL_POINTS))
COPDadm$FRAM_TOTAL_POINTS <- ftpPts$FTP



# indicator columns

indCol <- grep( "TOTAL", grep("_IND|_PRESCRIBED|_GT",
    names(COPDadm), value=T), invert=T, value=T )
statusCol <- which(colnames(COPDadm) %in% outcomes
    [[1]])

# set up models

# set.seed(1)
testFrac <- 0.2
tRows <- createDataPartition( COPDadm[, statusCol], p=
    testFrac, list=F )
```

```
# train on all training dataset patients, test on MG
    patients only

mgRows <- which( COPDadm[, "MG_PCP_24M_SEEN_IND"] == 1
     )
tMgRows <- tRows[tRows %in% mgRows]
nMgRows <- length( tMgRows )

COPDadm <- COPDadm[, ! names(COPDadm) %in% excludeCol]
dsfTestRows <- tMgRows
threshold <- 0.05
oma <- c(0, 0, 3, 0)
period <- years(1)

# set up test dates

fwdTestDates <- forward.test.dates( keyDates, today,
    period )
backTestDates <- backward.test.dates( keyDates, today,
     period )
midTestDates <- mid.test.dates( keyDates, today,
    period )

baseText <- paste( "COPD model validation for dates 
    between", years[1], "and", years[length(years)] )

testModel <- list(
  Full=
  list( Model=glm.test,
        ModelRowArg=dsfTestRows,
        TestRows=dsfTestRows,
        Text=
            paste( baseText, ", \n test =",
                   testFrac, "of total",
                   "sensitivity threshold (% of 
                      population", "flagged) =",
                   threshold ) ),

# set up filter arguments
```

```
rcnt <- c("_MR", "WEIGHT")
rcntCol <- unique( Reduce( c, sapply( rcnt, grep,
   colnames(COPDadm) ) ) )

minCor <- 0.6
sensThreshold <- 0.01
signLev <- 0.95

# run the test program

scenario <- "Full"

testRows <- testModel[[scenario]][["TestRows"]]

# use train set only

trainRows <- rownames( COPDadm[-testRows, ] )

# remove or backfill sparse columns

pctNA <- percent.NA.col( COPDadm[trainRows, rcntCol] )

# usable sparse columns

rareCol <- names( pctNA[pctNA <= SPARSE_MIN] )
if ( length(rareCol) > 0 ) {
  rareData <- na.to.colMeans(COPDadm[, rareCol])
  COPDadm[, rareCol] <- rareData
}

# data too sparse to use

sparseCol <- names( pctNA[pctNA > SPARSE_MIN] )
dataSet <- COPDadm[, ! colnames(COPDadm) %in%
   sparseCol]

# eliminate indicators with only one value

singleValCol <- which( sapply( dataSet[trainRows, ],
   function(c) length(unique(c)) ) <= 1 )
```

```
dataSet <- dataSet[, -singleValCol]
indCol <- indCol[indCol %in% colnames(dataSet)]

# indicator odds ratios

oRi <- odds.ratio.matrix(dataSet[trainRows, ], dataSet
    [trainRows, outcomes[[1]]], indCol, level=signLevel
    )

rnORbase <- rownames(oRi)
posCountMap <- rep( 1, length(rnORbase) )
names(posCountMap) <- rnORbase
posStat <- pos.stat( dataSet[trainRows, ], names(
    posCountMap), outcomes[[1]], posCountMap )

oddsRatioInd <- merge( oRi, posStat, by="row.names",
    all.x=T, rownames=T)
rownames(oddsRatioInd) <- oddsRatioInd$Row.names
outputCol <- c("IndPosCount", "IndPosPct", "
    IndPosOutcomePct", "Odds.Ratio", "CI.Lower", "CI.
    Upper", "Pr...z..", "Validity")

total <- data.frame(lapply(outputCol, function(c) NA))
colnames(total) <- outputCol
total <-
    mutate( total,
             Row.names="TOTAL",
             IndPosCount=nrow(dataSet[trainRows, ]),
             IndPosOutcomePct=
                 nrow( dataSet[trainRows, ]
                     [dataSet[trainRows, outcomes
                         [[1]]]==1,] ) /
                     nrow(dataSet[trainRows, ]) )
oRind <- rbind(oddsRatioInd[, c("Row.names", outputCol
    )], total)
colnames(oRind)[colnames(oRind) == "Row.names"] <- "
    Variable"
write.csv( oRind, paste(interimDir, "OR_", timeStamp,
    ".csv", sep=""), row.names=F )
```

```
indSignif <- na.omit(oRind[oRind$Validity == '*', "
    Variable"])

# numeric odds ratio

nc <- colnames(dataSet[, !colnames(dataSet) %in% c(
    indCol, "PAT_MRN_ID", outcomes[[1]])])
numCol <- nc[sapply( dataSet[, nc], is.numeric )]
oddsRatioNum <-
    odds.ratio.num( dataSet[trainRows, ],
                    dataSet[trainRows, outcomes[[1]]],
                    numCol,
                    level=signLev )

# combined univariate ratio

uniOutCol <- c( "Row.names", "Odds.Ratio", "Pr...z..",
     "CI.Lower", "CI.Upper", "Validity" )
oRn <- transform( oddsRatioNum, Row.names=rownames(
    oddsRatioNum) )

# univariate ethnicity

ethnicitySummary <-
  multi.category.stats( dataSet[trainRows, ],
                        "RACE_ETHNICITY",
                        outcomes[[1]],
                        sensThreshold,
                        "Caucasian",
                        "Other",
                        c("0", "1") )
write.csv( ethnicitySummary, paste( interimDir, "
    Ethnicity_", timeStamp, ".csv", sep="" ), row.names
    =F )


levels(dataSet$RACE_ETHNICITY) <- c(levels(dataSet$
    RACE_ETHNICITY), "AfroEuroAsian")
dataSet[! dataSet$RACE_ETHNICITY %in% c("Asian","
    African American", "Caucasian"), "RACE_ETHNICITY"]
    <- "Other"

dataSet[dataSet$RACE_ETHNICITY %in% c("Asian","African
    American", "Caucasian"), "RACE_ETHNICITY"] <- "
    AfroEuroAsian"


# marital status

maritalStatus <- multi.category.stats( dataSet[
    trainRows, ], "MARITAL_STATUS", outcomes[[1]],
    sensThreshold, "Married",  "Unknown", c("1", "0") )
write.csv( maritalStatus, paste(interimDir, "
    MaritalStatus_", timeStamp, ".csv", sep=""), row.
    names=F )

levels(dataSet$MARITAL_STATUS) <- c(levels(dataSet$
    MARITAL_STATUS), "NotWidowed", "Other")
dataSet[! dataSet$MARITAL_STATUS %in% c("Married","
    Widowed"), "MARITAL_STATUS"] <- "Other"
dataSet[dataSet$MARITAL_STATUS %in% c("Married","Other
    "), "MARITAL_STATUS"] <- "NotWidowed"

eTs <- transform( subset( ethnicitySummary, ! RACE_
    ETHNICITY %in% c( "TOTAL", "Caucasian" ) ),
Row.names=paste( RACE_ETHNICITY, ": Caucasian" ) )
mSt <- transform( subset( maritalStatus, ! MARITAL_
    STATUS %in% c( "TOTAL", "Married" ) ), Row.names=
    paste( MARITAL_STATUS, ": Married" ) )
uniOddRatio <- rbind( oddsRatioInd[, uniOutCol], oRn[,
     uniOutCol], eTs[, uniOutCol], mSt[, uniOutCol] )
colnames(uniOddRatio)[colnames(uniOddRatio) == "Row.
    names"] <- "Variable"
write.csv( uniOddRatio, paste(interimDir, "UniOR_",
    timeStamp, ".csv", sep=""), row.names=F )

cDs <- colnames(dataSet)
hiCorPriorNum <- high.corr( dataSet[trainRows, sort(
    cDs[which( !cDs %in% indCol )] )], minCor, "
    Numerical factor correlation matrix")
```

```
write.csv( hiCorPriorNum, paste( interimDir, "CorVar_
    Prior_", timeStamp, ".csv", sep="" ), row.names=F )
hiCorNumDf <- high.corr.df( hiCorPriorNum, minCor )
write.csv( hiCorNumDf, paste( interimDir, "CorVar_
    PriorTable_", timeStamp, ".csv", sep="" ), row.
    names=F )

hiCorPriorInd <- high.corr( dataSet[trainRows, sort(
    indCol)], minCor, "Indicator correlation matrix",
    scaleArg=list( cex=0.75 ) )
write.csv( hiCorPriorInd, paste( interimDir, "
    CorVarInd_", timeStamp, ".csv", sep="" ), row.names
    =F )
hiCorIndDf <- high.corr.df( hiCorPriorInd, minCor )
write.csv( hiCorIndDf,paste( interimDir, "CorVar_
    PriorIndTable_", timeStamp, ".csv", sep="" ), row.
    names=F )
indSignif <- indSignif[!indSignif %in% hiCorIndDf$var2
    ]

exclCol <- c(
  "MG_PCP_18M_SEEN_IND"
   , "LOOPDIURETIC_PRESCRIBED"
   , "NOT_PRN_MED_TOTAL_PRESCRIBED"
   , "DAYS_SINCE_LAST_EF"
   , "DISCHARGED_DISP_31_365_DAYS"
   , "NUM_HOSP_31_365_DAYS_COPD"
   , "NUM_ER_VISITS_31_365_DAYS_COPD"
   , "NUM_HOSP_31_365_DAYS_PNEU"
   , "NUM_ER_VISITS_31_365_DAYS_PNEU"
   , "TOTAL_LOS_HOSP_DAYS_LST_12_MO"
   , "NUM_HOSP_365_DAYS_COPD"
   , "NUM_ER_VISITS_365_DAYS_COPD"
   , "EJFR_NUM"
)
inclCol <- c( c(
  "PAT_AGE_YRS",
  "CARDO_24MM_VISIT",
  "PULMO_24MM_VISIT",
  "HOSP_30D_VISIT",
  "NUM_HOSP_365_DAYS_COPD",
  "NUM_ER_VISITS_365_DAYS_COPD",
  "NUM_ER_VISITS_365_DAYS_PNEU",
  "HOSP_12M_VISIT",
  "NUM_UNIQUE_SPECIALTIES",
  outcomes[[1]],
  numCol, indCol
) )
factorCol <- unique( subset( inclCol, ! inclCol %in%
   exclCol ) )
hiCor <- high.corr( dataSet[trainRows, sort( factorCol
    )], minCor, "Factor correlation matrix", scaleArg=
   c( cex=0.6 ) )
if ( Reduce( '*', dim(hiCor) ) > 0 ) {
  hiCorIndDf <- high.corr.df( hiCor, minCor )
}

train <- dataSet[, factorCol]
statusCol <- which(colnames(train) %in% outcomes[[1]])

# test on the "odds ratio" training dataset

fm <- glm("INP_OBS_COPD_ADM_365_DAYS ~ .", train[
   trainRows,], family='binomial', maxit=100, trace=T)
trainGlm <- auc.perf( fm, train[trainRows, ], train[
   trainRows, statusCol], type='response', text='
   Training set AUC' )

# test on "odds ratio" test dataset  selection

fullModel <- glm.test( train, statusCol, testRows,
   threshold, trace=T, maxit=100, text=testModel[[
   scenario]][["Text"]], oma=c(0,0,3,0), varJoiner='+'
    )
coefOut <- odds.ratio.stat( fullModel$model,
   uniOddRatio, signLev )
write.csv( coefOut, paste( interimDir, "OR_All_",
   timeStamp, ".csv", sep="" ), row.names=F )
coefSummary <- coef( summary( fullModel$model ) )
write.csv( coefSummary[order( rownames( coefSummary )
    ), ], paste( resultDir, "Coef_All_", timeStamp, ".
    csv", sep="" ) )

# generate performance statistics

prSort <- perf.clsf.stat( fullModel$perf$predicted,
    nrow( COPDadmRaw ), train[testRows, outcomes[[1]]]
    )

# plot the PPV / sensitivity graph

g <- ppv.sens.plot( prSort, main=baseText )
print( g )
ggsave( file=paste( "PPV_Sens_", scenario, "_",
    timeStamp, ".pdf", sep="" ), path=graphDir, plot=g,
     width=11, height=8.5, units="in" )

# two.ord.plot( prSort, main=testModel[[scenario]][["
    Text"]] )

allRisk <- cbind( PAT_MRN_ID=dataSet[rownames(prSort),
    "PAT_MRN_ID"], prSort )
write.csv( allRisk, paste( resultDir, "AllRisk_",
    timeStamp, ".csv", sep="" ), row.names=F )

# do hospitalizations matter?

testNull <- dataSet[testRows, ]
hospLastYr <- which( testNull$NUM_HOSP_365_DAYS_COPD >
     0 )
hospNextYr <-  which( testNull[, outcomes[[1]]] == 1 )
hospLastNextYr <- which( testNull[hospLastYr, outcomes
    [[1]]] == 1 )
numHospLastYr <- length( hospLastYr )
numHospNextYr <- length( hospNextYr )
numHospLastNextYr <- length( hospLastNextYr )
totalSize <- nrow( testNull )
nullHyp <- mutate (
      data.frame( Total.Population.Size=nrow( testNull
          ),
      Num.Total.Hosp.Last.Yr=numHospLastYr,
      Num.Hosp.Next.Yr=numHospNextYr,
      Num.Hosp.Last.Next.Yr=numHospLastNextYr,
      Pct.Hosp.COPD.Last.Yr=numHospLastYr / totalSize,
      Pct.Hosp.COPD.Last.Next.Yr=numHospLastNextYr /
          numHospLastYr,
      Pct.Flagged.Hosp.Total=numHospLastNextYr /
          length( hospNextYr ) ),
      F1=2.0 / ( 1.0 / Pct.Hosp.COPD.Last.Next.Yr +
          1.0 / Pct.Flagged.Hosp.Total ) )

write.csv( nullHyp, paste( interimDir, "NullHyp_",
    timeStamp, ".csv", sep="" ), row.names=F )

# lift table

pctList <- sort( c( 0.01, 0.05, 0.1, 0.2, nullHyp$Pct.
    Hosp.COPD.Last.Yr, 0.3, 0.5, 1  ) )
liftTable <- lift.table( prSort, pctList, nrow(
    dataSet ) )
mostRecent <- which( COPDadmRaw$AS_OF_DATE == today )
liftTable <- transform( liftTable, Num.Flagged.Most.
    Recent=NS.CA.round( Pct.Total * length( mostRecent
    ) ) )
write.csv( liftTable, paste( interimDir, "Lift_",
    timeStamp, ".csv", sep="" ), row.names=F )

# top 5 %

liftThreshold <- 0.05
hiRisk <- head( allRisk, liftThreshold * nrow( prSort
    ) )
write.csv( hiRisk, paste( resultDir, "HiRisk_",
    timeStamp, ".csv", sep="" ), row.names=F )

# random 100 from top 5%

write.csv( hiRisk[sample( 1:nrow(hiRisk), 100 ), ],
    paste( resultDir, "HiRiskRand100_", timeStamp, ".
    csv", sep="" ), row.names=F)
```

Listing C.1: Example: R code for COPD logistic regression.

The most recent version of the code in Listing C.1 can be found in COPDadm.R.

Appendix D Sample code used to implement the Cox proportional hazard model in the heart failure mortality example

In the listing below, the formatting is different from that of the actual code due to a different line length in the typeset font. User-defined packages `NS.CA.statUtils`, `NS.CA.dataUtils`, `NS.CA.dateUtils`, `NS.CA.modelUtils`, `NS.CA.plotUtils` and `NS.CA.mathUtils` are defined in Appendix E.

```
### Daniel Chertok
### (C) 2014 NorthShore University HealthSystem
### All rights reserved.

rm(list=ls())

library(debug)
library(reshape2)
library(date)
library(stringr)
library(plyr)
library(scales)
library(lubridate)

library(NS.CA.statUtils)
library(NS.CA.dataUtils)
library(NS.CA.dateUtils)
library(NS.CA.modelUtils)
library(NS.CA.plotUtils)
library(NS.CA.mathUtils)

# main module
# set up constants
```

```
testFrac <- 0.2
threshold <- 0.05
oma <- c( 0, 0, 3, 0 )
period <- years( 1 )
minCor <- 0.6
sensThreshold <- 0.01
signLev <- 0.95

years <- 2010:2012
today <- as.Date("2013-01-01")
lastDate <- as.Date("2012-01-01")
dataDir <- "../Data/"
resultDir <- "../Results/"
graphDir <- paste( resultDir, "Graphs", sep="" )
interimDir <- paste( resultDir, "Interim/", sep="" )
timeStamp <- timestamp.from.date()

# read the input file

admRaw <- read.csv( paste(dataDir, "HF_EOL.csv", sep="
    ") )
outcomes <- c( "DEATH_365_DAYS" )
admRaw[, outcomes] <- as.factor( admRaw[, outcomes] )

# all dates will be used later

keyDates <- sort( unique( as.Date( admRaw$AS_OF_DATE )
     ) )

# for analysis, use data up to today only

admRaw <- transform( admRaw, AS_OF_DATE=as.Date( AS_OF
    _DATE ) )
adm <- subset( admRaw, AS_OF_DATE <= lastDate )

# columns irrelevant to the analysis

excludeCol <- c( "PAT_ID" )
```

```
# columns where NAs can be turne into 0s

NAzero <- c("IND", "PRESCRIBED", "VISIT", "DAYS", "NUM
    " )
NAzeroCol <- unique( Reduce( c, sapply( NAzero, grep,
    colnames( adm ) ) ) )
NAzCol <- NAzeroCol[! colnames( adm[, NAzeroCol] ) %in
    % outcomes]
adm[, NAzCol] <- sapply( adm[, NAzCol], function( x )
   ifelse( is.na( x ), 0, x ) )

# independent of train / test breakdown

ejfrMed <- median( na.omit( admRaw$EJFR_NUM ) )
adm <- mutate( adm, EJFR_IND=ifelse( EJFR_NUM > 0, 1,
    0 ),
                EJFR_NUM=ifelse( EJFR_NUM > 0, EJFR_NUM
                    , ejfrMed ),
                EJFR_DIF=abs( EJFR_NUM - ejfrMed ),
                FEMALE_IND=ifelse( GENDER_CODE=="F", 1,
                    0 ) )
adm[is.na( adm$MARITAL_STATUS ),"MARITAL_STATUS"] <- "
    Unknown"
adm[is.na( adm$RACE_ETHNICITY ),"RACE_ETHNICITY"] <-
   NS.CA.mode( adm$RACE_ETHNICITY )

# indicator columns

indCol <- grep( "TOTAL", grep( "_IND|_PRESCRIBED|_GT",
     names( adm ), value=T ),
                invert=T, value=T )
statusCol <- which( colnames( adm ) %in% outcomes )

# select random rows for survival

idCol <- "PAT_MRN_ID"
patRows <- c( rand.surv( adm[adm$AS_OF_DATE < lastDate
    , ], idCol, seed=1 ),
              rownames( adm[adm$AS_OF_DATE == lastDate
                  , ] ) )
```

```
adm <- adm[patRows, ! colnames( adm ) %in% excludeCol]

# set.seed( NULL )   # random sampling
# set.seed( 1 )   # reproducible results
# testRows <- createDataPartition( adm[, statusCol], p
    =testFrac, list=F )
# testRows <- which( adm$AS_OF_DATE < as.Date(
    '2011-01-01' )) # test 2010-2011
testRows <- which( adm$AS_OF_DATE == lastDate )  #
    test on MG 2012-01-01
# testRows <- which(adm$AS_OF_DATE == '2014-01-01')  #
      test on MG 2014-01-01

# set up filter arguments

rcnt <- c( "__MR", "__06_MO", "__12_MO", "__24_MO", "_
    _36_MO", "__48_MO" )
rcntCol <- unique( Reduce( c, sapply( rcnt, grep,
    colnames( adm ) ) ) )

# use train set only

trainRows <- rownames( adm[-testRows, ] )

# remove or backfill sparse columns

dataSet <- rem.sparse.col( adm, trainRows, rcntCol,
    0.2 )

# convert outcomes to numeric

dataSet[, outcomes] <-
  as.numeric( levels( dataSet[, outcomes]  ) )[dataSet
      [, outcomes] ]

# eliminate indicators with only one value

dataTrain <- rem.single.val.col( dataSet[trainRows, ]
    )
dataTrainCol <- colnames( dataTrain )
```

```
dataSet <- dataSet[, dataTrainCol]
indCol <- indCol[indCol %in% dataTrainCol]


# univariate ethnicity

ethnicitySummary <-
  multi.category.stats( dataTrain, "RACE_ETHNICITY",
     outcomes, sensThreshold,
                       "Caucasian", "Other", c("0", "
                          1") )
write.csv( ethnicitySummary, paste( interimDir, "
   Ethnicity_", timeStamp,
                                   ".csv", sep="" ),
                                      row.names=F )

# this comes in after the initial analysis

levels(dataSet$RACE_ETHNICITY) <- c(levels(dataSet$
   RACE_ETHNICITY),
                                   "AfroEuroAsian")
dataSet[! dataSet$RACE_ETHNICITY %in%
         c("Asian","African_American", "Caucasian"),
            "RACE_ETHNICITY"] <-
  "Other"
dataSet[dataSet$RACE_ETHNICITY %in%
         c("Asian","African_American", "Caucasian"),
            "RACE_ETHNICITY"] <-
  "AfroEuroAsian"

# marital status

maritalStatus <-
  multi.category.stats( dataTrain, "MARITAL_STATUS",
     outcomes, sensThreshold,
                       "Married",  "Unknown", c( "1",
                          "0" ) )
write.csv( maritalStatus, paste( interimDir, "
   MaritalStatus_", timeStamp,
                                 ".csv", sep="" ),
                                        row.names=F )
levels(dataSet$MARITAL_STATUS) <- c( levels( dataSet$
   MARITAL_STATUS ),
                                        "NotWidowed", "
                                           Other" )
dataSet[! dataSet$MARITAL_STATUS %in% c( "Married","
   Widowed" ),
        "MARITAL_STATUS"] <- "Other"
dataSet[dataSet$MARITAL_STATUS %in% c( "Married","
   Other" ),
        "MARITAL_STATUS"] <- "NotWidowed"

eTs <- transform( subset( ethnicitySummary,
                          ! RACE_ETHNICITY %in% c( "
                             TOTAL", "Caucasian" ) ),
                  Variable=paste( RACE_ETHNICITY, ": 
                     Caucasian" ) )
mSt <- transform( subset( maritalStatus,
                          ! MARITAL_STATUS %in% c( "
                             TOTAL", "Married" ) ),
                  Variable=paste( MARITAL_STATUS, ": 
                     Married" ) )

uniOddsRatio <- odds.ratio.save( dataTrain, outcomes,
   indCol, idCol,
                                 addList=list( eTs,
                                    mSt ), oRdir=
                                    interimDir )

outCol <- which( colnames( dataTrain ) %in% outcomes )
hiCorTab <- output.corr( dataTrain, indCol, dir=
   interimDir,
                         timeStamp=timeStamp )

indSignif <- na.omit( uniOddsRatio[uniOddsRatio$
   Validity == '*', "Variable"] )
indSignif <- indSignif[!indSignif %in% as.character(
   hiCorTab$var2 )]

# manual adjustments to predictive variables
```

```
exclCol <- c( "ADN_EVER"     # correlated variables
            , "BMI__06_MO"
            , "BMI__12_MO"
            , "WEIGHT__MR"
            , "WEIGHT__06_MO"
            , "WEIGHT__12_MO"
            , "WEIGHT__24_MO"
            , "NUM_CARD_SEEN_MOST_LST_24_MO"
            , "VISIT_PRACTICE_NAME"
            , "LDL__MR"
            , "CREAT__06_MO"
            , "DBP__06_MO"
            , "SBP__06_MO"
            , "NUM_KCC_VISIT"
            , "MR_VISIT_PRIM_SPEC"
            , "NOT_PRN_MED_TOTAL_PRESCRIBED"
            , "PACEMAKER_IND"
            , "VISIT_PHY_MG_IND"
            , "NONINSULIN_DIAB_PRESCRIBED"

            , "CANCER_DX"
            , "CHOL__MR"
            , "COMBOANTHYP_PRESCRIBED"
            , "DBP__12_MO"
            , "DBP__24_MO"
            , "SBP__12_MO"
            , "DISCHARGED_DISP_30_DAYS"
            , "DISCHARGED_DISP_365_DAYS"
            , "EJFR_IND"
            , "EJFR_DIF"
            , "EJFR_NUM"
            , "INSULIN_PRESCRIBED"
            , "MG_ONC_VISIT"
            , "NUM_HOSP_30_DAYS_CHF"
            , "NUM_HOSP_365_DAYS_CHF"
            , "NUM_MISSED_APPTS_365"
            , "ORALVASODILSPERF_PRESCRIBED"
            , "PLAT__MR"
            , "MR_PHY_ID"
```

```
                    , "ACE_INHIBITOR_PRESCRIBED"
                    , "SPIRON_PRESCRIBED"
                    , "NUM_HOSP_365_DAYS_ICU"
                    , "LOOPDIURETIC_PRESCRIBED"
                    , "ASPIRIN_PRESCRIBED"
                    , "CARDO_12MM_VISIT"

                    , "INP_OBS_HF_365_DAYS"      # two look-
                       ahead variables, can't use
                    , "INP_OBS_HF_ADM_365_DAYS"

                    , "ANTICOAG_PRESCRIBED"      # latest
                       iteration testing on 2012
                    , "FEMALE_IND"
                    , "MI_IND"
                    , "PL_DEPR_IND"
                    , "METOLAZONE_PRESCRIBED"
                    , "THIAZIDEDIUR_PRESCRIBED"
                    , "TRIGL__MR"
                    , "NUM_HOSP_30_DAYS"
                   #, "NUM_HOSP_365_DAYS"
                    , "COPD_IND"
                    , "NUM_MISSED_APPTS_3_YRS"
                    , "HDL__MR"
)
inclCol <- c( outcomes[[1]]
                    , indSignif[indSignif %in% colnames(
                       dataSet )]
                    , "BETA_BLOCKER_PRESCRIBED"
                    , "CSO_EVER"     # latest iteration
                       testing on 2012
                   #, "CARDO_24MM_VISIT"
                    , "SBP__MR"
)

baseText <- paste( "HF_EOL model validation for dates
   between", years[1], "and",
                      years[length(years)] )
testText <- paste( baseText, ", \n test =", testFrac,
   "of total",
                    "sensitivity␣threshold␣(%␣of␣
                        population", "flagged)␣=",
                    threshold )

prSort <- run.glm.model( dataSet, trainRows, testRows,
     inclCol, exclCol,
                           outcomes, uniOddsRatio,
                              length( patRows ),
                           testText=testText, oRdir=
                              interimDir,
                           coefDir=resultDir, varJoiner=
                              '+', timeStamp=timeStamp )

# plot the PPV / sensitivity graph

scenario <- "Full"
allRisk <- ppv.sens.risk( dataSet, prSort, scenario,
    baseText,
                            graphDir=graphDir, resultDir
                               =resultDir,
                            timeStamp=timeStamp )

# lift table, top 5% and random 100 from top 5%

lift.table.save( prSort, nrow( dataSet ), admRaw$AS_OF
    _DATE,
                  max( admRaw$AS_OF_DATE ), allRisk,
                  c( 0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 1
                     ),
                  resultDir=resultDir, timeStamp=
                     timeStamp )

# Cox proportional hazard model

prSortCox <- run.cox.model( dataTrain, dataSet[
    testRows, ], today, inclCol,
                              exclCol, outcomes, coefDir
                                 =resultDir,
                              timeStamp=timeStamp )
```

```
baseTextCox <- paste( "HF␣EOL␣Cox␣survival␣model␣
    validation␣for␣dates␣between",
                        years[1], "and", years[length(
                            years)] )
allRiskCox <- ppv.sens.risk( dataSet, prSortCox,
    scenario, baseTextCox,
                            graphDir=graphDir,
                                resultDir=resultDir,
                            ppvFile="PPV_Sens_Cox_",
                                riskFile="AllRisk_Cox_
                                ",
                            timeStamp=timeStamp  )

# lift table, top 5% and random 100 from top 5%

lift.table.save( prSortCox, nrow( dataSet ), admRaw$AS
    _OF_DATE,
                max( admRaw$AS_OF_DATE ), allRiskCox,
                c( 0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 1
                    ),
                resultDir=resultDir, liftFile="Lift_
                    Cox_",
                riskFile="HiRisk_Cox_", topRiskFile="
                    HiRiskRand_Cox_",
                timeStamp=timeStamp )
```

Listing D.1: Example: R code for Cox proportional hazard modeling.

The most recent version of the code in Listing D.1 can be found in HF_-EOL.R.

Appendix E R package manuals

E.1 Package NS.CA.dataUtils - data manipulation

E.2 Package NS.CA.dateUtils - date arithmetic

E.3 Package NS.CA.mathUtils - math functions (re)defined

E.4 Package NS.CA.modelUtils - model performance descriptors and graphics

E.5 Package NS.CA.plotUtils - plotting utilities

E.6 Package NS.CA.statUtils - statistical functions

Package 'NS.CA.dataUtils'

April 12, 2016

Type Package

Title Manipulate, backfill and transform data

Version 3.0

Date 2015-01-28

Author Daniel Chertok (This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.)

Maintainer Daniel Chertok <`dchertok@northshore.org`>

Description Contains functions for backfilling, transforming and manipulating data for the purposes of predictive analytics.

Depends R (>= 3.0.3)

Imports stringr, dplyr, zoo, NS.CA.statUtils

License GPL-3

LazyData true

RoxygenNote 5.0.1

R topics documented:

`apply.na.omit` *Apply a function to all values that are not empty, NAs for empty values*

Description

Apply a function to all values that are not empty, NAs for empty values

Usage

```
apply.na.omit(x, i, fn, ...)
```

Arguments

`x`	data frame or matrix
`i`	dimension of application: 1 - rows, 2 columns (see "apply")
`fn`	function to apply
`...`	parameters passed to fn

Value

A list of resulting function applications

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`bind.with.field` *Add a field to non-null elements of a data frame*

Description

Add a field to non-null elements of a data frame

Usage

```
bind.with.field(x, field, c)
```

Arguments

`x`	data frame of dates
`field`	name of the field to add
`c`	field value(s)

Value

The original data frame with the field added

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

case.sentence — *String to sentence case (first capital) or from sentence case to lower case*

Description

String to sentence case (first capital) or from sentence case to lower case

Usage

```
case.sentence(x, inverse = T)
```

Arguments

`x`	string to convert
`inverse`	if TRUE, convert `x` to sentence case, otherwise, to lower case

Value

original string in sentence or lower case

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

df.to.zoo — *Convert a data frame to zoo*

Description

Convert a data frame to zoo

Usage

```
df.to.zoo(x, indCol = c("date", "hour"), format = "%F %H", tz = "UTC")
```

Arguments

`x`	data frame
`indCol`	time-generating columns of `x` (defaults to `c( "date", "hour" )`)
`format`	format for converting `indCol` to time (defaults to `"%F %H"`)
`tz`	time zone for time (defaults to `"UTC"`)

Value

zoo object generated from x

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

---

factor.to.numeric *Convert factor data to numeric*

Description

Convert factor data to numeric

Usage

```
factor.to.numeric(x)
```

Arguments

x	factor data

Value

original factor data converted to numeric

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

---

group.by.col *Group data frame by columns using group_by*

Description

Group data frame by columns using group_by

Usage

```
group.by.col(x, groupCol)
```

Arguments

x	data frame
groupCol	names of columns of x by which to group

Value

original data frame grouped by columns

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`list.invert` *Swaps names and values in an isomorphic list*

Description

In an isomorphic (1:1, or bijection) list, inverts the relationship turning names into values and values into names

Usage

```
list.invert(l)
```

Arguments

`l`	named list

Value

original list indexed by values with values set to original names

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`list.search` *Find entries in a list*

Description

Finds names of element(s) of a named list element by value

Usage

```
list.search(l, c)
```

Arguments

`l`	named list
`c`	- value to find in `l`

Value

names of element(s) of `l` whose values are equal to `c`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`map.direct` *Hash table (map)*

Description

Finds the value in a hash table (map) given by a mapping function corresponding to the given key

Usage

```
map.direct(key, map, ...)
```

Arguments

`key`	key to look for
`map`	mapping function
`...`	additional parameters passed to `map`

Value

value in the map corresponding to the supplied key

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`map.invert` *Key(s) from a map by value*

Description

Given a mapping (hash) function, finds keys corresponding to a given value by calling `list.invert`)

Usage

```
map.invert(value, mapFn, ...)
```

Arguments

`value`	value to find in the map
`mapFn`	mapping function
`...`	additional parameters passed to `mapFn`

Value

name(s) in the map corresponding to the supplied value

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`na.to.col.mode` *Replace NA's with column modes*

Description

Replace NA's with column modes

Usage

```
na.to.col.mode(x)
```

Arguments

`x`	matrix or data frame containing sparse data; convert x to matrix if it isn't

Value

The original matrix or data frame with NA's filled

`na.to.col.mode.2` *Replace NA's with column modes (corrected `na.to.col.mode`)*

Description

Replace NA's with column modes (corrected `na.to.col.mode`)

Usage

```
na.to.col.mode.2(x, fn = function(c) max(c, na.rm = T), ...)
```

Arguments

`x`	matrix or data frame containing sparse data; convert x to matrix if it isn't
`fn`	tiebreaker function in case of multimodality (defaults to `max( c, na.rm=T )`)
`...`	additional parameters passed to `fn`

Value

The original matrix or data frame with NA's filled

`na.to.colMeans` *Replace NA's with column means*

Description

Replace NA's with column means

Usage

```
na.to.colMeans(x)
```

Arguments

`x`	matrix or data frame containing sparse data; convert x to matrix if it isn't

Value

The original matrix or data frame with NA's filled

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`na.to.most.recent` *Replace NA's with the most recent data from prior months*

Description

Replace NA's with the most recent data from prior months

Usage

```
na.to.most.recent(x, valid, timeStr, mrStr = "...MR", funMR = function(y) {
     y })
```

Arguments

`x`	matrix or data frame containing sparse data
`valid`	valid numerical time points in field names
`timeStr`	time string pattern in field names identifying time intervals (e.g., "\.Mo$")
`mrStr`	appended to newly created field names to indicate most recent data
`funMR`	function to use for postprocessing the the data (e.g., replacing NA's; defaults to identity function)

Value

The original matrix or data frame with "most recent" columns added

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`na.to.row.mode` *Replace NA's with row modes*

Description

Replace NA's with row modes

Usage

```
na.to.row.mode(x)
```

Arguments

`x`	matrix or data frame containing sparse data; convert x to matrix if it isn't

Value

The original matrix or data frame with NA's filled

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`na.to.rowMeans` *Replace NA's with row means*

Description

Replace NA's with row means

Usage

```
na.to.rowMeans(x)
```

Arguments

`x`	matrix or data frame containing sparse data; convert x to matrix if it isn't

Value

The original matrix or data frame with NA's filled

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`NS.CA.dataUtils` *NS.CA.dataUtils.*

Description

Data manipulation, transformation and imputation

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`pattern.to.str` *Replace a pattern with a string*

Description

Replace a pattern with a string

Usage

```
pattern.to.str(x, pattern, str)
```

Arguments

`x`	matrix or data frame containing sparse data
`pattern`	what to replace
`str`	replacement

Value

The original with pattern changed to str

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`remove.col.by.name` *Remove column(s) by name or number or matrix*

Description

Remove column(s) by name or number or matrix

Usage

```
remove.col.by.name(lx, field, isStr)
```

Arguments

`lx`	list of data frames or matrices
`field`	- name of the field to add
`isStr`	(logical) is this a column name?

Value

original (list of) data frame(s) with the field(s) removed

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

Index

Package ‘NS.CA.dateUtils’

April 12, 2016

Type Package

Title Date manipulation and arithmetic

Version 5.0

Date 2015-01-28

Author Daniel Chertok (This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.)

Maintainer Daniel Chertok <dchertok@northshore.org>

Description
Contains functions for manipulating and analyzing dates for the purposes of predictive analytics.

Depends R (>= 3.0.3)

Imports timeDate, lubridate, NS.CA.dataUtils

License GPL-3

LazyData true

RoxygenNote 5.0.1

R topics documented:

`date.hr.seq` *Date and hour time marks*

Description

Generates an ordered sequence of dates and corresponding time marks

Usage

```
date.hr.seq(dateFrom, dateTo, seqFn = hours24, colNames = c("date", "hour"),
  ...)
```

Arguments

`dateFrom`	start date of the sequence
`dateTo`	end date of the sequence
`seqFn`	function generating daily time marks (defaults to `hours.24`)
`colNames`	column names of the resulting data frame (defaults to `c( "date", "hour" )`)
`...`	further arguments passed to `seqFn`

Value

Returns a data frame with dates as the first column and subday time marks as subsequent columns ordered right to left (e.g., by hour, then by date)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`date.ind` *Date to date components*

Description

Appends the following columns to a data frame containing a date:

day of the week

day of the year

month

hour

year

date index in sequence

ISO date (YYYY-MM-DD)

Usage

```
date.ind(x, col)
```

Arguments

`x`	data frame containing a date column
`col`	date column in `x`

Value

Returns the original data frame with date element columns appended

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`date.to.POSIXct` *convert* `Date` *to* `POSIXct`

Description

convert `Date` to `POSIXct`

Usage

```
date.to.POSIXct(d, format = "%F", tz = "UTC")
```

Arguments

`d`	Date
`format`	format of `d` (defaults to "%F")
`tz`	time zone (defaults to "UTC")

Value

Returns a `POSIXct` object

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`days.in.year.Date` *Calculate the # of days in a given year forward or backward*

Description

Calculate the # of days in a given year forward or backward

Usage

```
days.in.year.Date(dt, direction = 1)
```

Arguments

`dt`	current date
`direction`	count forward (>0) or backward(<0)

Value

(integer) # of days in the year

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`dec.to.hr.str` *Decimal timestamp to a string containing hours, minutes and seconds*

Description

Converts a decimal date in the form YYYYMMDD.hhmmss to a string in the form of "hh hr(s) mm min ss sec"

Usage

```
dec.to.hr.str(x)
```

Arguments

`x`	timestamp in decimal form

Value

Returns a string in the form of "hh hr(s) mm min ss sec"

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`dow.map` *Day-of-the week map*

Description

Generates a map of days of the week to their three-letter names

Usage

```
dow.map()
```

Value

Returns a map in the form of "3-letter day name" -> day # starting at Sun=0

`first.date` *Earliest date by row from data frame columns*

Description

Earliest date by row from data frame columns

Usage

```
first.date(x, cutoff = Sys.Date() - as.difftime(6240, units = "weeks"))
```

Arguments

`x`	data frame of dates
`cutoff`	dates before that time are set to the minimum date defaults to Sys.Date() - 120 years

Value

Returns a list of earliest dates in each row

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`first.valid.date` *First non-na date by row from data frame columns*

Description

First non-na date by row from data frame columns

Usage

```
first.valid.date(x, dateNow = as.Date("1970-01-01"), op = `<=`, fn = min)
```

Arguments

`x`	data frame of dates
`dateNow`	cutoff date
`op`	comparison operator for cutoff
`fn`	aggregate function for cutoff

Value

Returns a list of earliest valid dates in each row

`hours` *Difference between two timestamps in hours*

Description

Difference between two timestamps in hours

Usage

```
hours(t1, t2)
```

Arguments

`t1`	start date
`t2`	end date

Value

Returns the difference between `t1` and `t2` in hours

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`hours24` *Hours of the day*

Description

Hours of the day

Usage

```
hours24()
```

Value

Returns whole numbers from 0 to 23

`mark.add` *Intelligently add to time mark*

Description

Intelligently add to time mark

Usage

```
mark.add(m, dm, mSet = mark.hr())
```

Arguments

`m`	time mark
`dm`	marks to ad to `m`
`mSet`	available marks (defaults to the result of `mark.hr()`)

Value

Returns a time mark `=24*wday( d ) + hour( d ); 0 <= mark <= 167`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`mark.from.date` *Convert a date to hourly time mark (by calling `mark.from.date.hr`)*

Description

Convert a date to hourly time mark (by calling `mark.from.date.hr`)

Usage

```
mark.from.date(d)
```

Arguments

`d`	date

Value

Returns a time mark `=24*wday( d ) + hour( d ); 0 <= mark <= 167`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`mark.from.date.hr`	*Convert date and hour to hourly time mark (by calling* `mark.from.day.hr`*)*

Description

Convert date and hour to hourly time mark (by calling `mark.from.day.hr`)

Usage

```
mark.from.date.hr(d, h)
```

Arguments

`d`	date
`h`	hour

Value

Returns a time mark `=24 * ( wday( d ) - 1 ) + h; 0 <= mark <= 167`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`mark.from.day.hr`	*Convert numeric weekday and hour to hourly time mark*

Description

Convert numeric weekday and hour to hourly time mark

Usage

```
mark.from.day.hr(d, h)
```

Arguments

`d`	(numeric) day of the week
`h`	hour

Value

Returns a time mark `=24*d + h; 0 <= mark <= 167`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`mark.hr` *Hourly time marks*

Description

Generates a sequence of integer time marks corresponding to hours of the day by day of the week

Usage

```
mark.hr(dow = 0:6, hr = 0:23)
```

Arguments

`dow`	vector of weekday #'s (defaults to 0:6, where "0" is Sunday)
`hr`	vector of hours of the day (defaults to 0:23)

Value

Returns a list of nonnegative whole numbers signifying consecutive hour marks for the given hours of given days of the week (defaults to 0:23)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`NS.CA.dateUtils` *NS.CA.dateUtils.*

Description

Date manipulation and date arithmetic

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`random.date` *Generate random dates within a given range*

Description

Generate random dates within a given range

Usage

```
random.date(dateFrom, dateTo, startFrac = 0, n = 1)
```

Arguments

`dateFrom`	interval start
`dateTo`	interval end
`startFrac`	starting point of the interval (between 0 and 1; defaults to 0)
`n`	number of dates to generate (defaults to 1)

Value

Returns `n` random dates between `dateFrom + ( dateTo - dateFrom ) * startFrac` and `dateTo`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`season.calendar.map` *Month-to-season map*

Description

Generates a map of months to the names of calendar seasons

Usage

```
season.calendar.map()
```

Value

Returns a map in the form of "season name" -> vector of month #'s

`season.from.month` *Season from month*

Description

Season from month

Usage

```
season.from.month(month)
```

Arguments

`month`	month(s) of the year (can be a list)

Value

Returns a list of seasons corresponding to `month(s)`

`season.subset` *Get entries from the given season*

Description

Given the name of a season, get the rows of a data that belong to this season

Usage

```
season.subset(x, season, field = "month", seasonMap = season.calendar.map)
```

Arguments

`x`	data frame
`season`	name of the season
`field`	name of the field containing the month of the corresponding date or timestamp
`seasonMap`	mapping function for seasons (defaults to `season.calendar.map`)))

Value

Returns the data frame containing only the rows from `season`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`timestamp.from.date` *Date to timestamp*

Description

Converts a date to timestamp in the form of [yyyy][sep][mm][sep][dd]

Usage

```
timestamp.from.date(d = Sys.Date(), sep = "_")
```

Arguments

`d`	date (defaults to `link{Sys.Date}`)
`sep`	year-month-day separator (defaults to "_")

Value

Returns a timestamp string in the form of [yyyy][sep][mm][sep][dd]

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`weekdays` *One-letter weekday names*

Description

Generates a sequence of one-letter weekday names ("U" - "S") or a one-letter name corresponding to the # of a day of the week (0 - 6)

Usage

```
weekdays(x)
```

Arguments

`x`	(optional) # of a day of the week

Value

Returns:

`if no x is supplied,`
"U", "M", "T", "W", "R", "F", "S"

`if x is supplied,`
one-letter name of the `x`-th day of the week

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`year.frac.Date` *Calculate time interval between two dates as a fraction of the year*

Description

Calculate time interval between two dates as a fraction of the year

Usage

```
## S3 method for class 'frac.Date'
year(dt1, dt2)
```

Arguments

`dt1`	start date
`dt2`	end date

Value

(numeric) fraction of the year

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`yy.to.yyyy` *2-digit year to 4-digit year*

Description

2-digit year to 4-digit year

Usage

```
yy.to.yyyy(x, cutoff = 50)
```

Arguments

`x`	array of dates
`cutoff`	year of the century beyond which conversion is upward (i.e., "51" converts to 2051)

Value

Returns dates with 2-digit years converted to 4-digit years: years below cutoff are converted to the current century, otherwise go to previous century

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

Index

Package 'NS.CA.mathUtils'

April 12, 2016

Type Package

Title Math extension functions

Version 2.0

Date 2014-07-09

Author Daniel Chertok (This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.)

Maintainer Daniel Chertok <dchertok@northshore.org>

Description Math extension functions.

Depends R (>= 3.0.3)

License GPL-3

LazyData true

RoxygenNote 5.0.1

R topics documented:

`minmax.step` *Min to max sequence from a vector*

Description

From a numeric vector generates a sequence from `min` to `max` with a given step

Usage

```
minmax.step(x, step = 1)
```

Arguments

`x`	vector of numbers
`step`	step of the sequence (defaults to 1)

Value

returns a sequence of numbers from min(x) to max(x) spaced at `step`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`NS.CA.mathUtils` *NS.CA.mathUtils.*

Description

Auxiliary and mathematical functions redefined from base (e.g., NS.CA.mathUtils:round provides standard mathematical, not IEEE, rounding)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`NS.CA.round` *Mathematically correct rounding*

Description

Rounds numbers down when the lat digit is less than 5, up otherwise

Usage

```
NS.CA.round(x)
```

Arguments

`x`	real number

Value

Returns (integer) ceiling(x) if x ends in 0.5 or greater; floor(x) otherwise

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

Index

Package 'NS.CA.modelUtils'

April 12, 2016

Title predictive modeling and statistical analysis utilities

Version 8.0

Date 2014-12-10

Author Daniel Chertok (This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.)

Maintainer Daniel Chertok <dchertok@northshore.org>

Description This is a collection of functions for setting up tests, analyzing survival times, calculating the odd ratio matrix with statistics and analyzing model performance.

Depends R (>= 3.0.3)

Imports ROCR, lubridate, plyr, ggplot2, survival, NS.CA.dataUtils, NS.CA.mathUtils, NS.CA.statUtils, NS.CA.dateUtils, NS.CA.plotUtils

License GPL-3

LazyData true

RoxygenNote 5.0.1

R topics documented:

`auc.perf` *Model performance evaluation*

Description

Plots AUC, Matthews correlation coefficient and F1 score `auc.perf.base`)

Usage

```
auc.perf(model, test, status, type, ...)
```

Arguments

`model`	evaluated model
`test`	testing data set
`status`	observed (actual) outcomes
`type`	type of predicted response (probability, response etc.)
`...`	additional arguments passed to `auc.perf.base`

Value

Returns a list of model outputs:

`predicted`	predicted outcomes
`pred`	prediction object from ROCR
`perf`	tpr/fpr performance object from ROCR
`auc`	full AUC object
`mat`	Matthews correlation coefficient
`f1`	f1 score

`auc.perf.base` *Model performance evaluation*

Description

Plots AUC, Matthews correlation coefficient and F1 score

Usage

```
auc.perf.base(prediction, status, digits = PERF.DIGITS, text = "",
  mfrow = AUC.MFROW, oma = AUC.OMA, plotFile = NA, width = 11,
  height = 8.5)
```

Arguments

`prediction`	predicted data
`status`	observed (actual) outcomes
`digits`	# of decimal digits to display on the AUC graph (defaults to `PERF.DIGITS`))
`text`	graph caption (defaults to an empty string)
`mfrow`	graph panel parameters (defaults to `AUC.MFROW`)
`oma`	graph margin parameters (defaults to `AUC.OMA`)
`plotFile`	name of the file for saving PDF of the plot (defaults to NA, in which case nothing is saved)
`width`	width of the plot in inches (defaults to 11, ignored if `plotFile` is missing)
`height`	height of the plot in inches (defaults to 8.5, ignored if `plotFile` is missing)

Value

Returns a list of model outputs:

`pred`	prediction object from ROCR
`perf`	tpr/fpr performance object from ROCR
`auc`	full AUC object
`mat`	Matthews correlation coefficient
`f1`	f1 score

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`backward.test.dates` *Set up backward test program*

Description

Break up dates into training and testing datasets allowing for at least one observed period after the training date

Usage

```
backward.test.dates(dates, dateNow = Sys.Date(), period = AUC.PERIOD)
```

Arguments

`dates`	list of dates (for which data is available)
`dateNow`	today's date (defaults to system date)
`period`	(observation) period (defaults to `AUC.PERIOD`)

Value

Returns a list of vectors of dates:

`Train`	for the training data set(first period)
`Test`	for all other periods at least one period prior to today

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`charge.save` *Charges for predicted hospitalizations*

Description

Compute and save charges for specific and general hospitalizations

Usage

```
charge.save(risk, x, chargeCol, idCol = ID.COL, predCol = PRED.COL,
  actCol = ACT.COL, chargeDir = ".", chargeFile = "Chrg_",
  timeStamp = TIMESTAMP)
```

Arguments

`risk`	data frame of computed and actual risk data ordered by predicted probabilities of outcome of interest in descending order. It should contain (at least) the following columns: **idCol** identification column (key; defaults to `ID.COL`) **predCol** predicted outcome column **actCol** actual outcome column
`x`	event data frame including charges
`chargeCol`	column(s) of charges in `x`
`idCol`	ID column of `x` (defaults to `ID.COL`)
`predCol`	column(s) of predicted probabilities of outcome of interest (defaults to `PRED.COL`)
`actCol`	column(s) of actual outcomes of interest (defaults to `ACT.COL`)
`chargeDir`	directory for saving the charge file (defaults to "./")
`chargeFile`	file name prefix for saving the charge file (defaults to "Chrg_")
`timeStamp`	file name identification timestamp (defaults to `TIMESTAMP`)

Value

Returns a data frame of mean charges per row of `x` (patient) with `chargeCol` columns

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`forward.test.dates` *Set up forward test program*

Description

Break up dates into training and testing datasets allowing for at least one observed period after the last test date

Usage

```
forward.test.dates(dates, dateNow = Sys.Date(), period = AUC.PERIOD)
```

Arguments

`dates`	list of dates (for which data is available)
`dateNow`	today's date (defaults to system date)
`period`	(observation) period (defaults to `AUC.PERIOD`)

Value

Returns a list of vectors of dates:

`Train`	for the training data set(first period)
`Test`	for all other periods at least one period prior to today

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`glm.test` *glm (classification) model fit assessment*

Description

Runs glm (classification) model fit assessments on a dataset

Usage

```
glm.test(dataSet, statusCol, testRows, threshold = AUC.THRESHOLD,
  family = "binomial", type = "response", trace = F, maxit = 25,
  varJoiner = "+", corrMat = F, ...)
```

Arguments

`dataSet`	orginal data set for testing the fit
`statusCol`	(observed) outcomes
`testRows`	rows belonging to the testing set
`threshold`	classification sensitivity threshold (i.e., what fraction of dataSet we are selecting as having an outome of interest; defaults to `AUC.THRESHOLD`)
`family`	model family (defaults to 'binomial')
`type`	response type for prediction (defaults to 'response')
`trace`	(logical) trace the convergence? (defaults to FALSE)
`maxit`	max # of iterations (defaults to 25)
`varJoiner`	variable joiner parameter: '+', '*' or ':' (defaults to '+')
`corrMat`	(logical) display correlation matrix plot (defaults to F)
`...`	additional arguments passed to `auc.perf`

Value

Returns a list of vectors of dates:

`model`	glm model
`eff`	predicted and actual outcomes side by side
`mean`	PPV of up to threshold highest ranking outcomes of interest
`perf`	model statistics (including auc) passed up from `auc.perf`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`glm.validate.time` *Validate glm (classification) model*

Description

Validates glm (classification) model according to the rules provided by a parameter function

Usage

```
glm.validate.time(dataSet, statusCol, testDates, threshold = AUC.THRESHOLD,
  dateField = "AS_OF_DATE", period = AUC.PERIOD, corrMat = F, ...)
```

Arguments

`dataSet`	orginal data set for testing the fit
`statusCol`	(observed) outcomes
`testDates`	Test dates within dataSet `AUC.PERIOD`), where: **dates** list of dates; **dateNow** date; **period** (lubridate) Period
`threshold`	classification sensitivity threshold (i.e., what fraction of dataSet we are selecting as having an outome of interest; defaults to `AUC.THRESHOLD`)
`dateField`	date filed in dataSet (defaults to "AS_OF_DATE")
`period`	(lubridate) Period between dataset dates (defaults to `AUC.PERIOD`)
`corrMat`	(logical) display correlation matrix plot (defaults to F)
`...`	additional arguments passed to `auc.perf`

Value

Returns a list of vectors of dates:

`model`	glm model

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

lift.table *Lift table*

Description

Tabulates PPV and sensitivity at specified points by percentage using linear approximation

Usage

```
lift.table(x, pctList, totalSize, pctCol = "Percent.Flagged",
  actPosCol = "Number.Actual", actCol = "Actual", ppvCol = "PPV",
  sensCol = "Sensitivity")
```

Arguments

`x`	model performance data frame
`pctList`	vector of data points (percentages) for which lift statistics are sought
`totalSize`	total population size
`pctCol`	column of `x` containing the percentage (flagged) column in `x` (defaults to "Percent.Flagged")
`actPosCol`	column of `x` containing the # of actual positive cases in the percentage of the test population given by `pctList` (defaults to "Number.Actual")
`actCol`	column of `x` containing the # of actual positive cases in the test population (defaults to "Number.Actual")
`ppvCol`	column of `x` containing positive predictive value (PPV) (defaults to "PPV")
`sensCol`	column of `x` containing sensitivity (defaults to "Sensitivity")

Value

Returns a data frame of lift statistics:

`Pct.Total`	percentage of total population for which lift statistics is sought
`Num.Flagged.Total`	# of entries in the total population corresponding to `Pct.Total`
`Num.Flagged.Test`	# of flagged entries in the test dataset corresponding to `Pct.Total`
`Num.True.Pos.Test`	# of true positives in the test dataset corresponding to `Pct.Total`
`PPV`	positive predicted value
`Sensitivity`	sensitivity
`F1`	F1 statistic (harmonic average of `PPV` and `Sensitivity`)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

lift.table.save — *Lift table and risk rankings for all and high-risk patients*

Description

Compute and save the lift table using `lift.table`) and save the risk matrices for all and `topRiskNum` high-risk patients

Usage

```
lift.table.save(prSort, nRec, dates, today, allRisk, steps = c(0.01, 0.05,
  0.1, 0.2, 0.3, 0.5, 1), nullHypStep = NA, resultDir = "./",
  liftFile = "Lift_", liftThreshold = 0.05, riskFile = "HiRisk_",
  topRiskNum = 100, topRiskFile = "HiRiskRand_", timeStamp = TIMESTAMP)
```

Arguments

`prSort`	data frame of risk statistics with row names corresponding to patient data entries in `x` computed as a result of `test.odds.ratio`)
`nRec`	total # of patients
`dates`	dates in the dataset
`today`	calculation date
`allRisk`	data frame of PPV / sensitivity / lift statistics computed as a result of `ppv.sens.risk`)
`steps`	tabulation steps of lift statistics table (defaults to c(0.01, 0.05, 0.1, 0.2, 0.3, 0.5, 1))
`nullHypStep`	tabulation step corresponding to the null hypothesis (benchmark algorithm; defaults to NA). If provided, is inserted into `steps` and sorted
`resultDir`	directory for saving the files (defaults to "./")
`liftFile`	file name prefix for saving lift statistics (defaults to "Lift_")
`liftThreshold`	fraction of patients from the list to select (defaults to 0.05)
`riskFile`	file name prefix for saving the risk rankings file (defaults to "HiRisk_")
`topRiskNum`	# of patients to select randomly from the high risk list (defaults to 100)
`topRiskFile`	file name prefix for saving the top risk rankings file (defaults to "HiRiskRand_topRiskNum")
`timeStamp`	file name identification timestamp (defaults to `TIMESTAMP`)

Value

No return parameter

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`mid.test.dates` *Set up backward test program*

Description

Break up dates into training and testing datasets allowing for at least one observed period after the training date

Usage

```
mid.test.dates(dates, dateNow = Sys.Date(), period = AUC.PERIOD)
```

Arguments

`dates`	list of dates (for which data is available)
`dateNow`	today's date (defaults to system date)
`period`	(observation) period (defaults to `AUC.PERIOD`)

Value

Returns a list of vectors of dates:

`Train`	for the training data set(first and last period)
`Test`	for all other periods in between

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`model.comp` *Compare two models by performance and charges*

Description

Compare performance of two models, compute and save respective patients' charges

Usage

```
model.comp(altPred, x, outCol, outInd, chargeCol, idCol = ID.COL,
  predCol = PRED.COL, nRows = 1000, aucText = "Original model",
  chargeDir = ".", chargeFile = "ChrgOM_", perfFile = "PerfOM_",
  timeStamp = TIMESTAMP, ...)
```

Arguments

`altPred`	data frame of computed pronbabilities of outcome of interest and actual outcome(s). It should contain (at least) the following columns: **idCol** identification column (key; defaults to `ID.COL`) **outcomes** outcome column(s)
`x`	event data frame including charges
`outCol`	vector of columns containing outcomes
`outInd`	outcome(s) of interest index(indices)
`chargeCol`	column(s) of charges in `x`
`idCol`	ID column of `x` (defaults to `ID.COL`)
`predCol`	column(s) of predicted probabilities of outcome of interest (defaults to `PRED.COL`)
`nRows`	number of entries in `x` (patients) selected for comparison
`aucText`	performance plot caption (defaults to "Original model")
`chargeDir`	directory for saving the charge file (defaults to "./")
`chargeFile`	file name prefix for saving the charge file (defaults to "ChrgOM_")
`perfFile`	file name prefix for saving the performance and charge file (defaults to "PerfOM_")
`timeStamp`	file name identification timestamp (defaults to `TIMESTAMP`)
`...`	additional arguments passed to `auc.perf.base`

Value

Returns a data frame of mean charges per row of `x` (patient) with the following columns

`outCol`	outcome column(s)
`chargeCol`	charge columns

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`NS.CA.modelUtils` *NS.CA.modelUtils.*

Description

Model performance statistics, test programming and graphing

Arguments

`AUC.MFROW`	c(2, 2)
`AUC.OMA`	c(0, 0, 2, 0)
`AUC.THRESHOLD`	0.05
`AUC.PERIOD`	<- years(1)
`SPARSE.MIN`	<- 0.2
`CORR.MIN`	<- 0.6
`SIG.LEV`	<- 0.95
`ID.COL`	<- "PAT_MRN_ID"
`PRED.COL`	<- "Predicted"
`ACT.COL`	<- "Actual"
`TIMESTAMP`	<- timestamp.from.date()
`COX.ITER.MAX`	<- 100
`PERF.DIGITS`	<- 4

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`odds.ratio.save` *Combined indicator and numerical odds ratio*

Description

Compute and save indicator and numerical odds ratios using `odds.ratio.summary`) and `odds.ratio.num`)

Usage

```
odds.ratio.save(x, statCol, indCol, idCol = ID.COL, addList = NA,
  oRdir = ".", indORfile = "ORind_", uniORfile = "ORuni_",
  sigLev = SIG.LEV, timeStamp = TIMESTAMP, ...)
```

Arguments

`x`	event data frame
`statCol`	name of the column of observed outcomes
`indCol`	indicator columns
`idCol`	ID column of x (defaults to `ID.COL`)
`addList`	list of additional odds ratio tables (defaults to NA)
`oRdir`	directory for saving the odds ratio files (defaults to "./")
`indORfile`	file name prefix for saving the indicator odds ratio file (defaults to "ORind_")
`uniORfile`	file name prefix for saving the univariate odds ratio file (defaults to "ORuni_")
`sigLev`	confidence level for testing the statistical significance of coefficients (their difference from 0; defaults to `SIG.LEV`))
`timeStamp`	file name identification timestamp (defaults to `TIMESTAMP`)
`...`	additional parameters passed to `odds.ratio.matrix`)

Value

Returns a data frame of univariate odds ratios:

`Variable`	variable name
`Odds.Ratio`	univariate odds ratio
`Std..Error`	standard error[ln(odds ratio)]
`Pr...z..`	probability of z.value exceeeding the significance level
`CI.Lower`	lower confidence interval boundary of odds ratio
`CI.Upper`	upper confidence interval boundary of odds ratio
`Validity`	statistical significance marker ('*' = valid)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`output.corr` *Plot and save correlation matrices and tables*

Description

Compute correlation matrices for indicator and numerical variables by calling `high.corr`) and `high.corr.df`) and save high correlation matrices to .csv files as data frames and tables

Usage

```
output.corr(x, indCol, minCor = CORR.MIN, dir = "./",
  corNumFile = "CorNum_Prior_", corNumTabFile = "CorNum_PriorTable_",
  corIndFile = "CorInd_Prior_", corIndTabFile = "CorInd_PriorTable_",
  corrPlotDir = "./", plotNumFile = "Corr_Num_",
  plotIndFile = "Corr_Ind_", timeStamp = TIMESTAMP, ...)
```

Arguments

`x`	event data frame
`indCol`	indicator columns
`minCor`	high correlation threshold (defaults to `CORR.MIN`)
`dir`	directory into which .csv files will be stored (defaults to "./")
`corNumFile`	file name prefix for numerical variable correlation matrix (defaults to "CorNum_Prior_")
`corNumTabFile`	file name prefix for numerical variable correlation table (defaults to "CorNum_PriorTable_")
`corIndFile`	file name prefix for indicator variable correlation matrix (defaults to "CorInd_Prior_")
`corIndTabFile`	file name prefix for indicator variable correlation matrix (defaults to "CorInd_PriorTable_")
`corrPlotDir`	directory for saving correlation matrix plots (defaults to "./")
`plotNumFile`	file name prefix for the numerical predictor correlation matrix (defaults to "Corr_Num_")
`plotIndFile`	file nam prefix for the indicator predictor correlation matrix (defaults to "Corr_ind_")
`timeStamp`	file name identification timestamp (defaults to `TIMESTAMP`)
`...`	additional arguments passed to `auc.perf.base`

Value

Returns a data frame of highly correlated pairs of variables, including their correlations:

`var1`	1st variable in a highly correlated pair
`var2`	2nd variable in a highly correlated pair
`cor`	correlation betwen the two variables

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`perf.clsf.stat` *Performance statistics for a classification model*

Description

Provides validation statistics for a classification model based on predicted (and, if availabel, actual) outcomes

Usage

```
perf.clsf.stat(predicted, totalSize = nrow(predicted), actual = NULL)
```

Arguments

`predicted`	predicted outcomes
`totalSize`	size of the original (whole) data set (defaults to `nrow( predicted )`
`actual`	Observed outcomes (defaults to NULL)

Value

Returns a data frame of model performance metrics sorted by predicted values in descending order with columns:
Regardless of whether outcomes in `actual` were supplied

`Predicted`	predicted values
`Number.Flagged`	# of entries flagged in the testing data set as having positive outcomes
`Number.In.Whole`	# of entries flagged in the whole data set (training + testing) as having positive outcomes

ONLY if outcomes in `actual` were supplied

`Actual`	(if actual is supplied) observed outcomes
`Number.Actual`	# of observed positive outcomes
`PPV`	positive predictive value
`Sensitivity`	model sensitivity
`Lift`	model lift at the given point

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`ppv.sens.risk` *Plot and save the PPV / sensitivity graph and the risk matrix*

Description

Plot and save the standard PPV / sensitivity graph `ppv.sens.plot`) and save the risk matrix

Usage

```
ppv.sens.risk(x, prSort, scenario, ppvSensText, idCol = ID.COL,
  graphDir = "./", resultDir = "./", ppvFile = "PPV_Sens_",
  riskFile = "AllRisk_", timeStamp = TIMESTAMP)
```

Arguments

`x`	event data frame
`prSort`	data frame of risk statistics with row names corresponding to patient data entries in `x` computed as a result of `test.odds.ratio`)
`scenario`	name of the scenario for which the data is saved
`ppvSensText`	graph title for the PPV / sensitivity plot
`idCol`	ID column of `x` (defaults to `ID.COL`)
`graphDir`	directory for saving the PPV / sensitivity plot (defaults to "./")
`resultDir`	directory for saving the risk file (defaults to "./")
`ppvFile`	file name prefix for saving the PPV / sensitivity plot (defaults to "PPVsens_")
`riskFile`	file name prefix for saving the risk file (defaults to "AllRisk_")
`timeStamp`	file name identification timestamp (defaults to `TIMESTAMP`)

Value

Returns a data frame of risk statistics with row names corresponding to patient data entries in `x` (see `perf.clsf.stat`).

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`rand.surv` *Random selection of survivor's data*

Description

Select one row of data for those patients in the mortality model who remained alive for at least two dates during the observation period. If patient has only one row, it is returned, otherwise one is selected at random from all patient's data.

Usage

```
rand.surv(x, field = ID.COL, seed = NULL)
```

Arguments

`x`	patient data frame
`field`	patient identifier (defaults to `ID.COL`)
`seed`	random sampler seed (defaults to NULL, (quasi)random)

Value

Returns a vector of row numbers

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`rem.single.val.col` *Remove single-valued columns from a data matrix*

Description

Remove single-valued (indicator) columns from a data frame

Usage

```
rem.single.val.col(x)
```

Arguments

`x`	event data frame

Value

Returns `x` without single-valued columns

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`rem.sparse.col` *Remove sparse columns from a data matrix*

Description

Remove from a data frame those columns where more than `SPARSE.MIN`) of the data is absent

Usage

```
rem.sparse.col(x, trainRows, col, sparseMin = SPARSE.MIN)
```

Arguments

`x`	event data frame
`trainRows`	rows of `x` used for the training dataset
`col`	columns of interest
`sparseMin`	threshold proportion of missing values above which a column is ignored

Value

Returns `x` without sparse rows

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`run.cox.model` *Run the* `coxph` *(Cox) model and compute its performance metrics*

Description

Run the Cox model and compute its performance metrics by calling `auc.perf`) and `perf.clsf.stat`)

Usage

```
run.cox.model(train, test, today, inclCol, exclCol, statCol,
  dDateCol = "DEATH_DATE", aDateCol = "AS_OF_DATE", varJoiner = "+",
  maxIter = COX.ITER.MAX, coefDir = ".", coefFile = "Coef_All_Cox_",
  plotDir = ".", plotFile = "Cox_AUC_",
  text = "Cox survival model analysis", timeStamp = TIMESTAMP, ...)
```

Arguments

`train`	training dataset
`test`	testing dataset
`today`	valuation date
`inclCol`	included columns
`exclCol`	excluded columns
`statCol`	name of the column of observed outcomes
`dDateCol`	outcome date column (defaults to "DEATH_DATE")
`aDateCol`	as-of date column (defaults to "AS_OF_DATE")
`varJoiner`	formula parameter for the `glm` model (defaults to '+')
`maxIter`	max # of iterations for `coxph` (defaults to `MAX.ITER`)
`coefDir`	directory for saving model coefficients (defaults to "./")
`coefFile`	file name prefix for saving model coefficients (defaults to "Coef_All_")
`plotDir`	directory for saving AUC plot files (defaults to "./")
`plotFile`	file name prefix for saving AUC plot files (defaults to "Cox_AUC_")
`text`	graph caption (defaults to "Cox survival model analysis")
`timeStamp`	file name identification timestamp (defaults to `TIMESTAMP`)
`...`	additional arguments passed to `auc.perf`

Value

Returns a data frame of risk statistics with row names corresponding to patient data entries in `x` (see `perf.clsf.stat`).

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`run.glm.model` *Run the `glm` model, compute correlation matrices and tables*

Description

Run the model, compute correlation matrices for indicator and numerical variables by calling `high.corr`) and `high.corr.df`), save high correlation matrices to .csv files as data frames and tables, save model coefficients as a table and compute and plot performance statistics using `perf.clsf.stat`)

Usage

```
run.glm.model(x, trainRows, testRows, inclCol, exclCol, statCol, uniOddsRatio,
  numRec, varJoiner = "+", minCor = CORR.MIN, scaleArg = c(cex = 0.6),
  threshold = 0.05, maxit = 100, sigLev = SIG.LEV, trace = T,
  trainText = "Training set AUC", testText = "Testing set AUC",
  oRdir = "./", oRfile = "OR_All_", coefDir = "./",
  coefFile = "Coef_All_", corrPlotDir = "./",
  trainAUCplotFile = "AUC_Train_", testAUCplotFile = "AUC_Test_",
  corrPlotFile = "Corr_Mat_All_", timeStamp = TIMESTAMP, ...)
```

Arguments

`x`	event data frame
`trainRows`	rows of `x` belonging to the training data set
`testRows`	rows of `x` belonging to the test data set
`inclCol`	included columns
`exclCol`	excluded columns
`statCol`	name of the column of observed outcomes
`uniOddsRatio`	univariate odds ratio data frame containing columns `Variable, Odds.Ratio, Pr...z.. , CI.Lowe` and `Validity` with row names the same as the values of `Variable`
`numRec`	total # of records in the original dataset
`varJoiner`	formula parameter for the `glm` model (defaults to '+')
`minCor`	high correlation threshold (defaults to `CORR.MIN`)
`scaleArg`	label scaling (defaults to c(cex=1), i.e., full size)
`threshold`	percentage of high risk patients to select (defaults to 0.05)
`maxit`	maximum number of iterations in the `glm` model (defaults to 100)
`sigLev`	confidence level for testing the statistical significance of coefficients (their difference from 0; defaults to `SIG.LEV`))
`trace`	trace `glm` iterations? (defaults to TRUE)
`trainText`	graph title for performance plots for the training set (defaults to 'Training set AUC')
`testText`	graph title for performance plots for the text set (defaults to "")
`oRdir`	directory for saving odds ratios (defaults to "./")
`oRfile`	file name prefix for saving odds ratios (defaults to "OR_All_")
`coefDir`	directory for saving model coefficients (defaults to "./")
`coefFile`	file name prefix for saving model coefficients (defaults to "Coef_All_")
`corrPlotDir`	directory into which correlation matrix plots will be stored (defaults to "./")
`trainAUCplotFile`	file name prefix for saving training set AUC plots (defaults to "AUC_Train_")
`testAUCplotFile`	file name prefix for saving testing set AUC plots (defaults to "AUC_Test_")
`corrPlotFile`	file name prefix for saving the all-factor correlation matrix plot (defaults to "Corr_Mat_All_")
`timeStamp`	file name identification timestamp (defaults to `TIMESTAMP`)
`...`	additional arguments passed to `auc.perf` and `high.corr`

Value

Returns a data frame of risk statistics with row names corresponding to patient data entries in `x` (see `perf.clsf.stat`).

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`surv.diff` *Survival time*

Description

Compute survival time as the number of days from the start of the observation interval to the minimum of outcome date and interval end

Usage

```
surv.diff(asOfDate, today, outcomeDate, fmt = "%m/%d/%Y")
```

Arguments

`asOfDate`	date stamp of the beginning of the interval
`today`	maximum survival date (today's date)
`outcomeDate`	date stamp of the end of the interval
`fmt`	date format (defaults to "%m/%d/%Y")

Value

Returns the difference in days (integer) between `asOfDate` and `min(outcomeDate, today)`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`surv.time.max` *Maximum survival time*

Description

Maximum survival time

Usage

```
surv.time.max(dt, startDate, maxDate, maxVal = 1e+10)
```

Arguments

`dt`	list or data frame of time intervals with possible NAs
`startDate`	list of interval starting dates
`maxDate`	scalar or list of cutoff dates
`maxVal`	maximum value for the interval (defaults to 1e10)

Value

Returns a list or data frame of intervals where NAs are backfilled with maximum observation lengths

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

Index

Package 'NS.CA.plotUtils'

April 12, 2016

Title plotting utilities

Version 3.0

Date 2015-01-28

Author Daniel Chertok (This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.)

Maintainer Daniel Chertok <dchertok@northshore.org>

Description Custom plotting functions.

Depends R (>= 3.0.3)

Imports plotrix, scales, ggplot2, NS.CA.dateUtils

License GPL-3

LazyData true

RoxygenNote 5.0.1

R topics documented:

`legend.quantile` *Quantile legend generator*

Description

Generates quantile legend

Usage

```
legend.quantile(quantile, shift = 1)
```

Arguments

`quantile`	vector of quantiles of interest (can be of length 1)
`shift`	how many intervals to exclude from the legend (defaults to 1; used mainly for compatibility with `ggplot`)

Value

Returns a list of strings describing quantiles

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`lm.eqn` *Construct a linear regression equation string from* `x` *and* `y` *using* `lm`

Description

Construct a linear regression equation string from `x` and `y` using `lm`

Usage

```
lm.eqn(x, y)
```

Arguments

`x`	x-values
`y`	y-values

Value

Returns a string containing the regression equation obtained using `lm( y ~ x )`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`NS.CA.plotUtils` *NS.CA.plotUtils*

Description

Functions for graphing (predictive) model performance metrics and statistics and functional dependencies for model data

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`plots.order` *Order a list of plots*

Description

Order a list of plots

Usage

```
plots.order(x, order)
```

Arguments

`x`	(named) list of plots
`order`	sort order

Value

Returns the original list of plots in the specified order

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`ppv.sens.plot` *PPV and Sensitivity plot*

Description

Plots a PPV and Sensitivity plot using ggplot

Usage

```
ppv.sens.plot(x, xCol = "Percent.Flagged", ppvCol = "PPV",
  sensCol = "Sensitivity", xTickSeq = seq(0, 1, by = 0.1),
  yTickSeq = xTickSeq, main = "Model performance metrics")
```

Arguments

`x`	model performance data frame
`xCol`	abscissa column in x (defaults to "Percent.Flagged")
`ppvCol`	positive predictive value (PPV) column in x (defaults to "PPV")
`sensCol`	sensitivity column (defaults to "Sensitivity")
`xTickSeq`	abscissa tick sequence (defaults to `seq( 0, 1, by=0.1 )`)
`yTickSeq`	ordinate tick sequence (defaults to `xTickSeq`)
`main`	plot title (defaults to "PPV / Sensitivity")
`...`	additional parameters passed to twoord.plot

Value

ggplot object

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`template.plot.hourly` *Temporal plot template*

Description

Temporal plot template

Usage

```
template.plot.hourly(x, breaks, xTick, legPos = "bottom", size = 1,
  colour = "black", linetype = "dotted", xAngle = 90)
```

Arguments

`x`	data frame to plot
`breaks`	tick mark positions
`xTick`	x-axis labels
`legPos`	legend position (defaults to 'bottom')
`size`	grid line size (defaults to 1)
`colour`	grid line color (defaults to black)
`linetype`	grid line type (defaults to "dotted")
`xAngle`	angle of rotation for x-axis labels (defaults to 90, counterclockwise)

Value

Returns a ggplot object with the specified parameters

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`theme.ribbon` *Create a* `ggplot` *theme for a ribbon plot*

Description

Create a `ggplot` theme for a ribbon plot

Usage

```
theme.ribbon(legPos = "bottom")
```

Arguments

`legPos`	legend position (deafults to 'bottom')

Value

Returns a theme for a ribbon plot

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`title.plot` *Plot title generator*

Description

Generates a plot title from the base title string, pavilion, season, date boundaries, quantile and modified acuity

Usage

```
title.plot(t, pavilion, season, dateFrom, dateTo, quantile, acute)
```

Arguments

`t`	base title string
`pavilion`	pavilion
`season`	season
`dateFrom`	starting date
`dateTo`	end date
`quantile`	vector of quantiles of interest
`acute`	modified acuity

Value

Returns an enhanced plot tile as a string

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`title.plot.elem` *Plot title font*

Description

Plot title font

Usage

```
title.plot.elem()
```

Value

Returns `element_text( size=12, colour = "black", face="bold",  vjust=0.12 )`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`two.ord.plot` *Two-ordinate PPV and Sensitivity plot*

Description

Plots a two-ordinate PPV and Sensitivity plot using plotrix:twoord.plot

Usage

```
two.ord.plot(x, xCol = "Percent.Flagged", ly = "PPV", ry = "Sensitivity",
  xlab = "Percentage of patients flagged", xTickSeq = seq(0, 1, by = 0.1),
  lyTickSeq = seq(0, 1, by = 0.05), ryTickSeq = xTickSeq,
  ylab = "Positive predictive value", rylab = "Sensitivity", lcol = "red",
  rcol = "blue", main = "PPV / Sensitivity", type = "l",
  do.first = "grid(ny=NA, col='black',lty='dotted')", ...)
```

Arguments

`x`	model performance data frame
`xCol`	abscissa column in x (defaults to "Percent.Flagged")
`ly`	left ordinate in x (defaults to "PPV")
`ry`	right ordinate in x (defaults to "Sensitivity")
`xlab`	asbcissa label (defaults to "Percentage of patients flagged")
`xTickSeq`	tick sequence (defaults to `seq( 0, 1, by=0.1 )`)
`lyTickSeq`	left oridante tick sequence (defaults to `seq( 0, 1, by=0.05 )`)
`ryTickSeq`	right oridante tick sequence (defaults to lyTickSeq)
`ylab`	left ordinate label (defaults to "Positive predictive value")
`rylab`	right ordinate label (defaults to "Sensitivity")
`lcol`	left ordinate color (defaults to "red")
`rcol`	right ordinate color (defaults to "blue")
`main`	plot title (defaults to "PPV / Sensitivity")
`type`	plot type (defaults to 'l')
`do.first`	plot oprations performed before plotting the curves (defaults to "grid(col='black',lty='dotted')")
`...`	additional parameters passed to twoord.plot

Value

None

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`x.mark.hourly` *X-axis tick placement for a weekly hour-by-hour plot*

Description

X-axis tick placement for a weekly hour-by-hour plot

Usage

```
x.mark.hourly(mark, xStep = 4)
```

Arguments

`mark`	original x-axis coordinates from `x`
`xStep`	tick interval (deafults to 4)

Value

Returns a vector of tick mark coordinates as an equidistant sequence

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`x.tick.hourly` *Tick marks for a weekly hour-by-hour plot*

Description

Tick marks for a weekly hour-by-hour plot

Usage

```
x.tick.hourly(xStep = 4)
```

Arguments

`xStep`	tick interval (deafults to 4)

Value

Returns a vector of tick marks in the form of "<one-letter day of the week>.<hour of the day out of 24>"

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

y.scale *Y axis scaling*

Description

Given a data frame, generates ticks for the y-axis and scales the graph using `y.tick`

Usage

```
y.scale(x, col, scale, step, minCol = "qDiffMin", maxCol = "qDiffMax",
  absOut = "abs", diffOut = "diff", outNames = c(absOut, diffOut))
```

Arguments

`x`	data frame
`col`	column of `x` to be graphed
`scale`	scaling vector in the form of c(min, max) to be passed to `ggplot` as y-axis limits
`step`	tick step
`minCol`	column of `x` containing minimum differences (defaults to `qDiffmin`)
`maxCol`	column of `x` containing maximum differences (defaults to `qDiffMax`)
`absOut`	name of the output element of the scale list containing absolute values (defaults to `abs`)
`diffOut`	name of the output element of the scale list containing differences from the benchmark (defaults to `diff`)
`outNames`	vector of names for the output element `scale` (defaults to `c( absOut, diffOut )`)

Value

Returns

`yTickSeq`	tick sequence for the y-axis
`scale`	list of two-element vectors of min and max absolute values and differences

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`y.tick` *Y axis ticks*

Description

Given a data vector, generates a sequence from min to max with a given step

Usage

```
y.tick(y, yStep, round = T)
```

Arguments

`y`	numeric vector
`yStep`	tick step
`round`	data rounding flag (defaults to T)

Value

Returns a sequence of #'s from `min(y)` to `max(y)` with a step of `yStep`

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

Index

Package ‘NS.CA.statUtils’

April 12, 2016

Title custom statistical utilities

Version 7.0

Date 2015-10-20

Author Daniel Chertok (This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.)

Maintainer Daniel Chertok <`dchertok@northshore.org`>

Description Custom statistical functions.

Depends R (>= 3.0.3)

Imports lattice, vcd, plyr, reshape2, ROCR

License GPL-3

LazyData true

RoxygenNote 5.0.1

R topics documented:

`high.corr`	*High correlation elements*

Description

Plots the correlation matrix of a dataset, returns high correlation elements

Usage

```
high.corr(dataSet, minCor, text = "Correlation matrix", xArg = list(x =
  list(rot = 90)), scaleArg = c(cex = 1), plotFile = NA, width = 11,
  height = 8.5, fnCor = abs, ...)
```

Arguments

`dataSet`	orginal data set for testing the fit
`minCor`	lower threshold for high correlation elements (defaults to 0.6)
`text`	plot caption
`xArg`	x label arguments (defaults to `list( x=list( rot=90 ) )`, i.e., x labels are rotated 90 degrees counterclockwise)
`scaleArg`	label scaling (defaults to c(cex=1), i.e., full size)
`plotFile`	file name prefix saving PDF of the plot (defaults to NA, in which case nothing is saved)
`width`	width of the plot in inches (defaults to 11, ignored if `plotFile` is missing)
`height`	height of the plot in inches (defaults to 8.5, ignored if `plotFile` is missing)
`fnCor`	function that defines low correlation threshold (defaults to `abs`)
`...`	additional parameters to pass to levelplot (correlation matrix plotting function)

Value

Returns a matrix with off-diagonal high correlation elements (other entries included)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`high.corr.df` *High correlation data frame*

Description

Package the elements of a (high) correlation matrix into a data frame

Usage

```
high.corr.df(hiCor, minCor, fnCor = abs)
```

Arguments

`hiCor`	(high) correlation matrix
`minCor`	lower threshold for high correlation elements (defaults to 0.6)
`fnCor`	function that defines low correlation threshold (defaults to `abs`)

Value

Returns a data frame containing:

`var1`	first variable in a high correlation pair
`var2`	second variable in a high correlation pair
`cor`	(high) correlation value

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`matt.corr` *Matthews correlation coefficient for large numbers*

Description

Matthews correlation coefficient for large numbers

Usage

```
matt.corr(pred)
```

Arguments

`pred`	ROCR::prediction object

Value

Mathews correlation coefficient

multi.category.stats *Relative benchmark contingency table statistics for categorical variables*

Description

For a categorical variable,

- select a benchmark level against which the rest will be measured
- create a 2 x 2 pairwise contingency table for the remaining levels against the benchmark
- calculate χ^2 p-values and frequencies of outcomes of interest

Usage

```
multi.category.stats(x, multiLevCol, outcome, rareThreshold, benchmark,
  multiLevColOther = "Other", outcomeCol = c("1", "0"),
  fnSignif = fisher.test, level = 0.95, ...)
```

Arguments

`x`	orginal data set (matrix or data frame)
`multiLevCol`	column of x containing the categorical variable used in the calculation of the statistics
`outcome`	name of column of x contating the output
`rareThreshold`	fraction of total below which categories are rolled up to the next level
`benchmark`	name of the benchmark level
`multiLevColOther`	name of the catgch-all category into which categories below rareThreshold are rolled up (defaults to "Other")
`outcomeCol`	column names for outcome columns (defaults to c("0", "1"))
`fnSignif`	function to use for significance testing (defaults to fisher.test)
`level`	rejection level for the CI (defaults to 0.95 or 5%)
`...`	additional parameters to pass to `odds.ratio.table`

Value

Returns a data frame whose columns contain:

`<multiLevelCol>`	levels of the original categorical variable
`<outcomeCol[1]>`	counts of variables at the first outcome level
`<outcomeCol[2]>`	counts of variables at the second outcome level
`Pct`	percentage of all entries in x in the given category
`p.value`	χ^2 value that corresponds to the given category
`PctPos`	fraction of "positive" outcomes for the given category

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`NS.CA.mode` *Mode(s) of the distribution*

Description

Mode(s) of the distribution

Usage

```
NS.CA.mode(x, fun = function(y) {     y })
```

Arguments

`x`	matrix or data frame containing the distribution(s) (convert to matrix if list)
`fun`	function determining which mode to select in the multimodal case

Value

Returns the mode of the distribution

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`NS.CA.statUtils` *NS.CA.statUtils*

Description

Statistical utilities for overriding similarly named functions from other packages (e.g., NS.CA.mode) and functions for describing (predictive) model performance metrics and statistics

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`odds.ratio.logit` *Odds ratio statistics for (primarily) numerical variables given the \ (prebuilt) model*

Description

Generates from a prebuilt model a single row data frame with unadjusted and adjusted odds ratios (OR) and confidence intervals (CI) for OR, mean (exponential) and standard (ln) logistic multivariate model errors

Usage

```
odds.ratio.logit(glmFit, coefInd, level = 0.95)
```

Arguments

`glmFit`	(generalized linear) model
`coefInd`	index of the coefficient of the variable in glmFit
`level`	rejection level for the CI (defaults to 0.95 or 5%)

Value

Returns a (single row) data frame with OR statistics as columns:

`Odds.Ratio`	odds ratio
`Std.Error`	standard error[ln(odds ratio)]
`P.Value`	P-value of [ln(odds ratio)]
`CI.Lower`	lower confidence interval boundary of odds ratio
`CI.Upper`	upper confidence interval boundary of odds ratio
`Validity`	statistical significance marker ('*' = valid)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`odds.ratio.matrix` *Odds ratio statistics*

Description

Generates from a data frame with row-wise outcomes a comprehensive odds ratio (OR) data frame that includes ln(odds ratio), confidence interval (CI) and p-value for ln(OR), OR proper, CI(OR) and the validity indicator (valid if 1 is not inside the CI)

Usage

```
odds.ratio.matrix(x, outcome, oddsRatioCol = colnames(x), level = 0.95, ...)
```

Arguments

`x`	orginal data set (matrix or data frame)
`outcome`	(vector of) outcomes
`oddsRatioCol`	columns to use for calculating OR (defaults to colnames(X))
`level`	rejection level for the CI (defaults to 0.95 or 5%)
`...`	additional parameters to pass to `odds.ratio.table`

Value

Returns a data frame with variable names as row names and ln(OR) and OR statistics as columns:

`Log.Odds.Ratio`	ln(odds ratio)
`Std..Error`	standard error[ln(odds ratio)]
`z.value`	z-score of ln(odds ratio)
`Pr...z..`	probability of z.value exceeeding the significance level
`Log.CI.Lower`	lower confidence interval boundary of ln(odds ratio)
`Log.CI.Upper`	upper confidence interval boundary ln(odds ratio)
`Odds.Ratio`	odds ratio
`CI.Lower`	lower confidence interval boundary of odds ratio
`CI.Upper`	upper confidence interval boundary of odds ratio
`Validity`	statistical significance marker ('*' = valid)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`odds.ratio.multi` *Odds ratio statistics for (primarily) numerical variables*

Description

Generates from a data frame with row-wise outcomes a comprehensive odds ratio data frame that includes the unadjusted and adjusted odds ratios (OR) and confidence intervals (CI) for OR, mean (exponential) and standard (ln) logistic univariate model errors

Usage

```
odds.ratio.multi(glmFit, level = 0.95)
```

Arguments

`glmFit`	fitted (GLM) model
`level`	rejection level for the CI (defaults to 0.95 or 5%)

Value

Returns a data frame with variable names as row names and OR and OR statistics as columns:

`Odds.Ratio`	odds ratio
`Std.Error`	standard error[ln(odds ratio)]
`CI.Lower`	lower confidence interval boundary of odds ratio
`CI.Upper`	upper confidence interval boundary of odds ratio
`Validity`	statistical significance marker ('*' = valid)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`odds.ratio.num` *Odds ratio statistics for (primarily) numerical variables*

Description

Generates from a data frame with row-wise outcomes a comprehensive odds ratio data frame that includes the unadjusted and adjusted odds ratios (OR) and confidence intervals (CI) for OR, mean (exponential) and standard (ln) logistic univariate model errors

Usage

```
odds.ratio.num(x, outcome, oddsRatioCol = colnames(x), level = 0.95)
```

Arguments

`x`	orginal data set (matrix or data frame)
`outcome`	(vector of) outcomes
`oddsRatioCol`	columns to use for calculating OR (defaults to colnames(X))
`level`	rejection level for the CI (defaults to 0.95 or 5%)

Value

Returns a data frame with variable names as row names and OR and OR statistics as columns:

`Odds.Ratio`	odds ratio
`Std.Error`	standard error[ln(odds ratio)]
`CI.Lower`	lower confidence interval boundary of odds ratio
`CI.Upper`	upper confidence interval boundary of odds ratio
`Validity`	statistical significance marker ('*' = valid)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

odds.ratio.stat — *Combine uni- and multivariate odds ratio statistics into one data frame*

Description

Merges a univariate odds ratio statistic with the multivariate one (calculated by odds.ratio.multi) and returns both as one data frame

Usage

```
odds.ratio.stat(glmFit, uniOddRatio, level = 0.95)
```

Arguments

`glmFit`	fitted (GLM) model
`uniOddRatio`	(a data frame of) univariate odds ratios
`level`	rejection level for the CI (defaults to 0.95 or 5%)

Value

Returns a data frame with variable names as row names and OR and OR statistics as columns:

`Variable`	Variable name
`Odds.Ratio.Uni`	univariate odds ratio
`CI.Lower.Uni`	lower confidence interval boundary of the univariate odds ratio
`CI.Upper.Uni`	upper confidence interval boundary of the univariate ratio
`Pr...z...Uni`	z score of the univariate odds ratio
`Validity.uni`	statistical significance marker of the univariate odds ratio ('*' = valid)
`Odds.Ratio.Multi`	multivariate odds ratio
`CI.Lower.Multi`	lower confidence interval boundary of the multivariate odds ratio
`CI.Upper.Multi`	upper confidence interval boundary of the multivariate odds ratio
`Pr...z...Multi`	z score of the multivariate odds ratio
`Validity.Multi`	statistical significance marker of the multivariate multivariate odds ratio ('*' = valid)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`odds.ratio.summary` *Construct a standardized odds ratio table with statistics*

Description

Standardize the ratio table created by `odds.ratio.matrix` to include only "IndPosCount", "IndPosPct", "IndPosOutcomePct", "Odds.Ratio", "CI.Lower", "CI.Upper", "Pr...z..", and "Validity"

Usage

```
odds.ratio.summary(oRi, x, statusCol)
```

Arguments

`oRi`	odds ratio matrix (output of `odds.ratio.matrix`)
`x`	event data frame
`statusCol`	outcome column

Value

Returns `x` without sparse rows

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`odds.ratio.table` *Odds ratio statistics*

Description

Generates from a contingency table a comprehensive odds ratio data frame that includes ln(odds ratio), confidence interval (CI) and p-value for ln(OR), OR proper, CI(OR) and the validity indicator (valid if 1 is not inside the CI).

Usage

```
odds.ratio.table(m, level = 0.95, lwr = "lwr", upr = "upr",
  oRat = "Log.Odds.Ratio")
```

Arguments

`m`	orginal data set (matrix or data frame)
`level`	rejection level for the CI (defaults to 0.95 or 5%)
`lwr`	column name of the lower boundary of the confidence interval (defaults to "lwr")
`upr`	column name of the upper boundary of the confidence interval (defaults to "upr")
`oRat`	column name of the log odds ratio (defaults to "Log.Odds.Ratio")

Value

Returns a data frame with variable names as row names and odds ratio (OR) statistics as columns:

`Log.Odds.Ratio`	ln(odds ratio)
`Std..Error`	standard error[ln(odds ratio)]
`z.value`	z-score of ln(odds ratio)
`Pr...z..`	probability of z.value exceeeding the significance level
`Log.CI.Lower`	lower confidence interval boundary of ln(odds ratio)
`Log.CI.Upper`	upper confidence interval boundary ln(odds ratio)
`Odds.Ratio`	odds ratio
`CI.Lower`	lower confidence interval boundary of odds ratio
`CI.Upper`	upper confidence interval boundary of odds ratio
`Validity`	statistical significance marker ('*' = valid)

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`percent.NA.col` *percentage of NAs in each column*

Description

percentage of NAs in each column

Usage

```
percent.NA.col(x)
```

Arguments

`x`	matrix or data frame

Value

Returns a 1 x ncol(x) matrix or data frame with percentages of NAs in x by column

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

`pos.stat` *Positive outcome statistics*

Description

For each specified element of the input data frame, computes for count and percentage of designated "positive" values, the # of positive outcomes, and the designated "positive" value of the element.

Usage

```
pos.stat(x, xNames, outcome, posCountMap, posOutcome = 1)
```

Arguments

`x`	orginal data set (matrix or data frame)
`xNames`	columns of x used in the calculation of statistics
`outcome`	name of column of x contating the output
`posCountMap`	map of desifganted "positive" values by parameter
`posOutcome`	designated outcome of interest (defaults to 1)

Value

Returns a data frame whose columns contain:

`IndPosCount`	# of elements of x whose value is designated "positive"
`IndPosPct`	percentage of "positive" parameter values as a fraction of total
`IndPosOutcomePct`	percentage of "positive" parameter values that correspond to positive outcomes in x
`PosInd`	designated "positive" value for each parameter

Note

This code is intended for education and information sharing purposes only. NorthShore University HealthSystem is not responsible for any unintended consequences resulting from the implementation thereof. Please use at your own risk.

Index

References

[1] Peter C. Austin and Jack V. Tu. "Automated variable selection methods for logistic regression produced unstable models for predicting acute myocardial infarction mortality". In: *Journal of Clinical Epidemiology* 57 (2004), 1138:1146. URL: http://scholar.google.com/scholar_url?hl=en&q=http://uncwddas.googlecode.com/files/article2.pdf&sa=X&scisig=AAGBfm1boy8t65iA3pQUDj39ob3C2r8UtQ&oi=scholarr.

[2] Rollin Brant. *Biostatistics II.* Mar. 24, 2004. URL: http://www.stat.ubc.ca/~rollin/teach/643w04/lec/.

[3] Aran Canes. "Identifying superutilizers in the Medicaid population: A simple and powerful technique". In: *Predictive Analytics World for Healthcare.* Boston, MA, Oct 6-7, 2014.

[4] Scott Chacon. *Pro Git (Expert's Voice in Software Development).* 2009. URL: http://git-scm.com/book.

[5] D.R. Cox. "Regression Models and Life-Tables". In: *Journal of the Royal Statistical Society. Series B (Methodological)* 34 (1972), 187:220.

[6] Chris J Date. *An introduction to database systems (7. ed.)* Addison-Wesley-Longman, 2000, pp. I–XXII, 1–938. ISBN: 978-0-201-68419-3.

[7] D. Diez. *Survival Analysis in R.* June 2013. URL: https://www.openintro.org/download.php?file=survival_analysis_in_R&referrer=/stat/surv.php.

[8] A. Doyle. *The Complete Sherlock Holmes (Knickerbocker Classics).* Race Point Publishing, 2013. ISBN: 8601404432897.

[9] Vincent Driessen. *A successful Git branching model.* Jan. 5, 2010. URL: http://nvie.com/posts/a-successful-git-branching-model/.

[10] A. Elixhauser et al. "Comorbidity Measures for Use with Administrative Data". In: *Medical Care* 1 (1998), 8:27.

[11] G.M. Fitzmaurice, N.M. Laird, and J.H. Ware. *Applied Longitudinal Analysis.* Wiley Series in Probability and Statistics. Wiley, 2012. ISBN: 9781118551790. URL: http://books.google.com/books?id=0exUN1yFBHEC.

[12] John Fox and Sanford Weisberg. *Cox proportional-hazard regression for survival data in R.* Feb. 23, 2011. URL: http://socserv.mcmaster.ca/jfox/Books/Companion/appendix/Appendix-Cox-Regression.pdf.

[13] B. French et al. *Regression modeling of longitudinal binary outcomes with outcome-dependent observation times*. May 13, 2014. URL: https://dbe.med.upenn.edu/biostat-research/sites/files/facultyfiles/french_cambridge_talk.pdf.

[14] G.H. Golub and C.F. Van Loan. *Matrix Computations*. Johns Hopkins Studies in the Mathematical Sciences. Johns Hopkins University Press, 2013. ISBN: 9781421407944. URL: http://books.google.com/books?id=X5YfsuCWpxMC.

[15] D.W. Hosmer, S. Lemeshow, and R.X. Sturdivant. *Applied Logistic Regression*. Wiley Series in Probability and Statistics. Wiley, 2013. ISBN: 9780470582473. URL: http://books.google.com/books?id=64JYAwAAQBAJ.

[16] Max Kuhn. "Building Predictive Models in R Using the caret Package". In: *Journal of Statistical Software* 28.5 (Nov. 2008), pp. 1–26. ISSN: 1548-7660. URL: http://www.jstatsoft.org/v28/i05.

[17] James Leeper and UCLA. *Choosing the Correct Statistical Test*. 2014. URL: http://www.ats.ucla.edu/stat/mult_pkg/whatstat/choosestat.html.

[18] Zhiguo Li Li et al. "Failure event prediction using the Cox proportional hazard model driven by frequent failure signatures". In: *IIE transactions* 39 (2007), 303:315. URL: http://mpac.engr.wisc.edu/pdf/paper27.pdf.

[19] K.-Y. Liang and S. Zeger. "Longitudinal Data Analysis Using Generalized Linear Models". In: *Biometrika* (1986), 13:22.

[20] Scott Mitchell. *The Eight Commandments of Source Code Control*. Nov. 8, 2008. URL: http://scottonwriting.net/sowblog/archive/2008/11/13/163320.aspx.

[21] D.C. Montgomery, E.A. Peck, and G.G. Vining. *Introduction to Linear Regression Analysis*. Wiley Series in Probability and Statistics. Wiley, 2012. ISBN: 9780470542811. URL: http://books.google.com/books?id=0yR4KUL4VDkC.

[22] D.S. Moore. *Essential Statistics*. W. H. Freeman, 2010. ISBN: 9781429931595. URL: http://books.google.com/books?id=Pf8jAAAAQBAJ.

[23] NIST. *NIST/SEMATECH e-Handbook of Statistical Methods*. Oct. 30, 2013. URL: http://www.itl.nist.gov/div898/handbook/index.htm.

[24] NPR. *The Secret Recordings of Carmen Segarra*. Sept. 26, 2014. URL: http://www.thisamericanlife.org/radio-archives/episode/536/the-secret-recordings-of-carmen-segarra.

[25] Therese Pigott. "A Review of Methods for Missing Data". In: *Educational Research and Evaluation* 7.4 (2001), 353:383. URL: http://www.stat.uchicago.edu/~eichler/stat24600/Admin/MissingDataReview.pdf.

[26] Maja Pohar and Janez Stare. "Making relative survival analysis relatively easy". In: *Computers in biology and medicine* 37 (2007), 1741:1749. URL: http://www.pauldickman.com/cancerepi/handouts/handouts_survival/Pohar2007.pdf.

[27] Maja Pohar Perme, Robin Henderson, and Janez Stare. "An approach to estimation in relative survival regression". In: *Biostatistics* 10.1 (2009), 136:146. URL: http://biostatistics.oxfordjournals.org/content/10/1/136.full.pdf+html.

[28] W.H. Press et al. *Numerical Recipes 3rd Edition: The Art of Scientific Computing*. Cambridge University Press, 2007. ISBN: 9780521880688. URL: http://apps.nrbook.com/empanel/index.html#.

[29] Rachel Schutt and Cathy O'Neil. *Doing Data Science: Straight Talk from the Frontline*. O'Reilly Media, Inc., 2013. ISBN: 1449358659, 9781449358655.

[30] D. Strauss et al. "An analytical method for longitudinal mortality studies". In: *Journal of insurance medicine* (2000), 217:225.

[31] D. Taeger and S. Kuhnt. *Statistical Hypothesis Testing with SAS and R*. Wiley, 2014. ISBN: 9781118762608. URL: http://books.google.com/books?id=mJa0AgAAQBAJ.

[32] W.M. Thorburn. *The Myth of Occam's Razor*. University Press, 1918. URL: http://books.google.com/books?id=1xP8NAAACAAJ.

[33] L.N. Trefethen and D. Bau. *Numerical Linear Algebra*. Society for Industrial and Applied Mathematics (SIAM, 3600 Market Street, Floor 6, Philadelphia, PA 19104), 1997. ISBN: 9780898719574. URL: http://books.google.com/books?id=JaPtxOytY7kC.

[34] S. Tzu and L. Giles. *The Art of War by Sun Tzu - Special Edition*. The Art of War - Special Edition. El Paso Norte Press. ISBN: 9781934255551. URL: http://books.google.com/books?id=T0HQOQVKWz8C.

[35] UCLA. *What statistical analysis should I use? Statistical analyses using Stata*. 2014. URL: http://www.ats.ucla.edu/stat/stata/whatstat/whatstat.htm#hsb.

[36] Joachim Vandekerckhove, Dora Matzke, and Eric-Jan Wagenmakers. *Model Comparison and the Principle of Parsimony*. 2014. URL: https://http://www.cidlab.com/prints/vandekerckhove2014model.pdf.

[37] X.H. Zhou, D.K. McClish, and N.A. Obuchowski. *Statistical Methods in Diagnostic Medicine*. second. Wiley Series in Probability and Statistics. Wiley, 2011. ISBN: 9780470183144. URL: http://books.google.com/books?id=NWGUtAdOT8kC.

Index