Practical 3A - Variable selection - Lasso

Louis OLIVE

Table of contents

1 Loading WRDS / Compustat / CRSP data with R **2**

2 Practical 3 **3**

3 Practical 3A - Lasso **10**
3.1 Task to perform . . . 10

4 Additional material **15**

5 The Logistic Regression model **16**
5.1 Informal introductory example . . . 16
5.2 A more formal definition . . . 24
5.3 Estimation . . . 27
5.3.1 Maximum Likelihood Estimation . . . 27
5.3.2 Numerical methods . . . 29
5.3.3 Logistic Regression as a machine learning approach . . . 37

6 Interpretation **38**
6.1 Coefficients . . . 39
6.2 Odds . . . 39
6.3 Odds ratio . . . 41

7 Inference **42**
7.1 Asymptotic properties of MLE . . . 42
7.2 Wald statistics . . . 43
7.2.1 Confidence intervals . . . 43
7.2.2 Tests on model coefficients . . . 46
7.3 Likelihood ratio tests . . . 49
7.4 Goodness of Fit test / Calibration . . . 51

8 Variable selection, model assessment **56**
8.1 Best subset . . . 56
8.2 Stepwise Logistic Regression . . . 58
8.3 Introducing penalized logistic regressions . . . 63
8.4 Model assessment . . . 70
8.4.1 Introductory example . . . 70
8.4.2 Hold-out approach . . . 79
8.4.3 K-fold Cross-Validation approach . . . 80

9 Case study **81**
9.1 EDA . . . 81
9.2 Performing PCA - A first Scoring function . . . 112
9.3 Coefficient interpretation . . . 125
9.4 Stepwise Logistic Regression . . . 128
9.5 Penalized Logistic Regression . . . 148
9.6 Model assessment . . . 152

10 References **165**

1 Loading WRDS / Compustat / CRSP data with R

We first load Accounting and Market data that we will use in this practical (see `1_wrds_data_wrangling.qmd` for more details on the data sources):

- Compustat Fundamental annual data:

```
# compustat_all <- readRDS("./wrds_data/compustat_all.rds")
```

- Compustat Company data:

```
# company_all <- readRDS("./wrds_data/company_all.rds")
```

- CRSP daily security file

```
# crsp_daily <- readRDS("./wrds_data/crsp_daily_full.rds")
```

Compustat and CRSP linking table:

```
# ccmxpf_linktable <- readRDS("./wrds_data/ccmxpf_linktable.rds")
```

2 Practical 3

The aim of the Practical 3 is to implement two different "off-the-shelf" methods.

In the Practical 1 and 2 we used "expert" insights from academics to build and assess models (eg. select specific predictors).

Here we are more agnostic, and re-using the set of predictors from the Practicals 1 and 2 (Altman ratios, Distance to default, Macro/Market predictors) completed with additional variables from `Practical 1 / Step 5` we want to fit the "best possible" models to predict bankruptcy.

This practical is intentionally less directed than the two first to encourage exploration and personal implementation and mimic a real life project.

We remind below the `Practical 1 / Step 5`:

Step 5: Feature Engineering, computing financial ratios from Fundamentals

Your task is to create additional predictors by computing financial ratios from Compustat Fundamentals data. This step prepares Practical 3.

Besides Altman ($X_1, ..., X_5$):

ent ratio-profiles. The final discriminant function is as follows:

$$\text{(I)} \quad Z = .012X_1 + .014X_2 + .033X_3 + .006X_4 + .999X_5$$

where X_1 = Working capital/Total assets
X_2 = Retained Earnings/Total assets
X_3 = Earnings before interest and taxes/Total assets
X_4 = Market value equity/Book value of total debt
X_5 = Sales/Total assets
Z = Overall Index

or Desbois (Desbois (2008)) financial ratios ($r_1, ..., r_{37}$) presented in the introductory case study:

The battery of economic and financial criteria comprises 22 ratios selected according to the following topics. Note that EBITDA refers to Earnings Before Interest, Taxes, Depreciation and Amortization.

Capitalization

r1 total debt / total assets;
r2 stockholders' equity / invested capital;
r3 short term debt / total debt;
r4 short term debt / total assets;
r5 long and medium term debt / total assets;

Weight of the debt

r6 total debt / gross product;
r7 long and medium term debt / gross product;
r8 short term debt / gross product;

Liquidity

r11 working capital / gross product;
r12 working capital / (real inputs - financial expenses);
r14 short term debt / circulating assets;

Debt servicing

r17 financial expenses / total debt;
r18 financial expenses / gross product;
r19 (financial expenses+ refunding of long and medium term capital) / gross product;
r21 financial expenses / EBITDA;
r22 (financial expenses + refunding of long and medium term capital)/EBITDA;

Capital profitability

r24 EBITDA /total assets;

Earnings

r28 EBITDA / gross product;
r30 available income / gross product;
r32 (EBITDA - financial expenses) / gross product;

Productive activity

r36 immobilized assets / gross product;
r37 gross product / total assets.

WRDS/Compustat provides 70 pre-calculated financial ratios for all U.S. companies (more details in document `wrds_data/WRDS_Industry_Financial_Ratio_Manual.pdf`):

WRDS Industry Financial Ratio

August 2016

WRDS Research Team

Overview

WRDS Industry Financial Ratio (WIFR hereafter) is a collection of most commonly used financial ratios by academic researchers. There are in total over 70 financial ratios grouped into the following seven categories: *Capitalization*, *Efficiency*, *Financial Soundness/Solvency*, *Liquidity*, *Profitability*, *Valuation* and *Others.* Ratios for each individual company as well as at industry aggregated level are included in the output.

As in Desbois, they are categorized by themes:

Financial Ratio Definition

<u>Theme Classification</u>

WIFR collects over 70 different financial ratios, categorized based on the economic intuition into the following seven groups:

1. **Capitalization**: measures the debt component of a firm's total capital structure, e.g.: Capitalization Ratio, Total Debt-to-Invested Capital Ratio;
2. **Efficiency**: captures the effectiveness of firm's usage of assets and liability, e.g.: Asset Turnover, Inventory Turnover;
3. **Financial Soundness/Solvency**: captures the firm's ability to meet long-term obligations, e.g.: Total Debt to Equity Ratio, Interest Coverage Ratio;
4. **Liquidity**: measures a firm's ability to meet its short-term obligations, e.g.: Current Ratio, Quick Ratio;
5. **Profitability**: measures the ability of a firm to generate profit, e.g.: ROA, Gross Profit Margin;
6. **Valuation**: estimates the attractiveness of a firm's stock (overpriced or underpriced), e.g.: P/E ratio, Shiller's CAPE ratio;
7. **Others**: Miscellaneous ratios, e.g.: R&D-to-Sales, Labor Expenses-to-Sales.

To produce these ratios, WRDS uses a SAS routine (it has been translated to R and was available online here, a local version is available in `wrds_data/wrds_ratio_github.R` for inspiration). In particular some ratios look at the evolution of Fundamentals between two or more reporting periods or fiscal years.

Ratio Definition/Construction:

We provide definition to each individual ratio in the **Appendix** section. Please refer to the previous discussion on Data Frequency and Outlier Control for general guideline on data alignment and other technical treatment.

For the ***underlying code*** used to produce these ratios, please refer to the "Financial Ratio SAS Code" section listed under "Manuals and Overviews" page.

See also the Appendix A of Tian et al. (2015) article (in folder `literature/lasso`):

Appendix A

In this appendix, we explain how we construct each candidate predictive variable using the CRSP and/or COMPUSTAT data items. We follow CHS (2008) in the construction of the variables EXCESS RETURN, SIGMA, PRICE, and MB. EXCESS RETURN is a firm's log excess return on its equity relative to that on the S&P 500 index. SIGMA is the standard deviation of the daily stock return observed over the previous three months. PRICE is the equity price per share truncated from the above at the value of $15 and then takes the logarithm. MB is the ratio of the market equity to the adjusted book equity to which we add a 10% difference between the market equity and book equity. All series are available to investors in real time. Below we provide details for the other 35 predictive variables:

$$\mathrm{ACTLCT} = \mathrm{ACT/LCT}; \mathrm{APSALE} = \mathrm{AP/SALE}; \mathrm{CASHAT} = \mathrm{CHE/AT};$$

$$\mathrm{CASHMTA} = \mathrm{CHE/(PRICE} * \mathrm{SHROUT} + \mathrm{LT} + \mathrm{MIB});$$

$$\mathrm{CHAT} = \mathrm{CH/AT}; \mathrm{CHLCT} = \mathrm{CH/LCT}; \mathrm{(EBIT} + \mathrm{DP)/AT} = \mathrm{(EBIT} + \mathrm{DP)/AT};$$

$$\mathrm{EBITAT} = \mathrm{EBIT/AT}; \mathrm{EBITSALE} = \mathrm{EBIT/SALE}; \mathrm{FAT} = \mathrm{(DLC} + 0.5 * \mathrm{DLTT)/AT};$$

$$\mathrm{FFOLT} = \mathrm{FFO/LT}; \mathrm{INVCHINVT} = \mathrm{INVCH/INVT}; \mathrm{INVTSALE} = \mathrm{INVT/SALE};$$

$$\mathrm{(LCT\text{-}CH)/AT} = \mathrm{(LCT\text{-}CH)/AT}; \mathrm{LCTAT} = \mathrm{LCT/AT}; \mathrm{LCTLT} = \mathrm{LCT/LT};$$

$$\mathrm{LCTSALE} = \mathrm{LCT/SALE}; \mathrm{LT/(LT} + \mathrm{MKET)} = \mathrm{LT/(LT} + \mathrm{MKET)}; \mathrm{LTAT} = \mathrm{LT/AT};$$

$$\mathrm{LTMTA} = \mathrm{LT/(PRICE} * \mathrm{SHROUT} + \mathrm{LT} + \mathrm{MIB)}; \mathrm{LOG(AT)} = \log(\mathrm{AT});$$

$$\mathrm{LOG(SALE)} = \log(\mathrm{abs(SALE)}); \mathrm{MVEF} = \mathrm{(abs(PRCC_F)} * \mathrm{CSHO)/(DLC} + 0.5 * \mathrm{DLTT});$$

$$\mathrm{NIAT} = \mathrm{NI/AT}; \mathrm{NIMTA} = \mathrm{NI/(PRICE} * \mathrm{SHROUT} + \mathrm{LT} + \mathrm{MIB)}; \mathrm{NISALE} = \mathrm{NI/SALE};$$

$$\begin{aligned} \text{OIADPAT} &= \text{OIADP/AT; OIADPSALE} = \text{OIADP/SALE; QALCT} \\ &= (\text{ACT} - \text{INVT})/\text{LCT;} \end{aligned}$$

$$\begin{aligned} \text{REAT} &= \text{RE/AT; RELCT} = \text{RE/LCT; RSIZE} \\ &= \log(\text{PRICE} * \text{SHROUT/TOTVAL}); \end{aligned}$$

$$\text{SALEAT} = \text{SALE/AT; SEQAT} = \text{SEQ/AT; WCAPAT} = \text{WCAP/AT.}$$

Task 1: create additional predictors by computing financial ratios from Compustat Fundamentals data.

TO DO Create a rather large (at least 20) set of additional ratios predictors (on top of Altman ratios, Distance to default, Macro/Market predictors). You can take inspiration from Desbois or Compustat ratios. If so, try to select your ratios in different "theme" categories (eg. Financial soundness, Debt servicing, Liquidity, Earnings/Profitability etc)

Briefly explore these new predictors:

- providing descriptive statistics and features distributions,
- showing/plotting individual features "interaction" with the response variable
- identifying "outliers" if any.

Visualizations help a lot.

You can perform if needed additional cleansing or handling (winsorizing) of outliers as suggested by WRDS/Compustat user guide:

Outlier Control:

> As ratio metrics often produce unintended extreme outliers, we impose two layers of outliers control before aggregating at the industry level. First, for all the monthly frequency firm level individual ratio results, we impose a winsorization at 1% level for extreme values, and truncate the outliers in the top and bottom percentile to be missing. Secondly, to arrive at the final ratio output, we enforce a 12 month moving average on the monthly frequency financial ratios. The second step serves two purpose: to further smooth the final output, and to fill in the truncated extreme months (from step 1) with firm-specific moving average.
>
> Note that the outlier controls are ***only*** applied to the ratios fed to the industry-level aggregation. Outputs for firm-level financial ratios are *raw* ratios without any truncation or smoothing. Hence researchers are advised to censor/smooth the raw ratios to get rid of the extreme outliers before conducting further analysis.

3 Practical 3A - Lasso

3.1 Task to perform

Task 2: Lasso for variable selection

The goal is to explore the usage of Lasso penalization on a bankruptcy data set as done in recent research papers Tian et al. (2015) and Chava et al. (2023) (see folder `literature/lasso`). It will allow you to understand deeper the R packages used to fit Lasso penalized Logistic regressions.

TO DO Use a Cross-Validation procedure adapted to the problem to select a best λ parameter for Lasso penalization on your bankruptcy data set.

You might start with a lighter version of your data, or event start to practice with the Desbois data set (see Exercise - case study at the end of the document).

Such analysis as been performed in Tian et al. (2015) using 39 bankruptcy predictors but the Cross-Validation procedure is not properly described.

However, in Chava et al. (2023), authors assess various methods as Lasso or Gradient Boosted Trees on a similar data set as ours to predict bankruptcy. They propose the following scheme to select models hyperparameters (among which the λ for Lasso penalization):

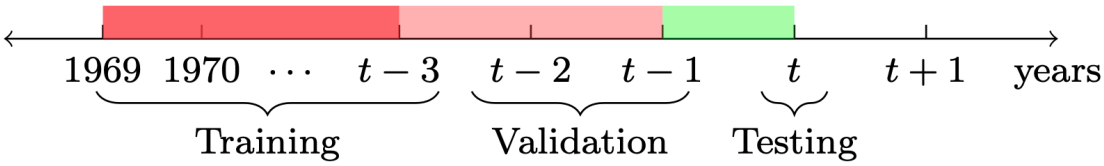
The common approach in the literature is to select tuning parameters adaptively from the data in a validation sample. In particular, we divide our sample into three disjoint time periods that maintain the temporal ordering of the data: training, validation, and out-of-sample testing.

Figure 5 presents the way we split our observations for every year in our sample for each ML method. In order to forecast default for year t in our sample, we first estimate a model subject to a specific set of tuning parameter values on a training sample which goes from 1969 (first year of our sample) to year $t-3$. In a second step we evaluate the set of hyperparameters by creating forecasts and computing the receiver operating characteristic area under the curve (AUC) in a 2 year validation sample of years $t-2$ and $t-1$. We iterate over a grid of hyperparameters in the first and second step and then select the set that minimizes the AUC in the validation sample. In a third step, we get the optimized model's prediction for year t, which was neither used for estimation nor tuning, so this is truly out-of-sample and provides an honest evaluation of a method's predictive performance.

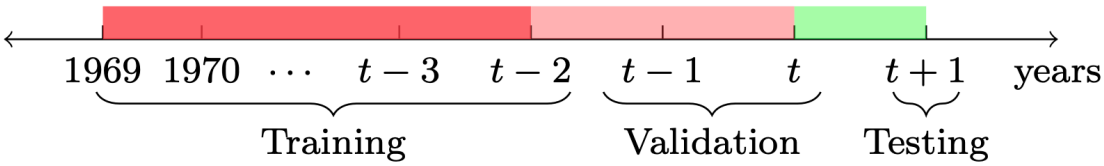
They use an expanding window (ie training set always uses data from data set starting year), but one might use as well a rolling window (ie a fixed horizon for training set, say 10Y):

As shown in Panel B of Figure 5, to forecast year $t+1$ we expand the training window by one year to now also include year $t-2$, we still use two years for validation (years $t-1$ and t), and we forecast year $t+1$. Thus, we follow an expanding window on our training set and always use the two prior years for validation.

Figure 5: Expanding Window for Training/Validation/Testing.

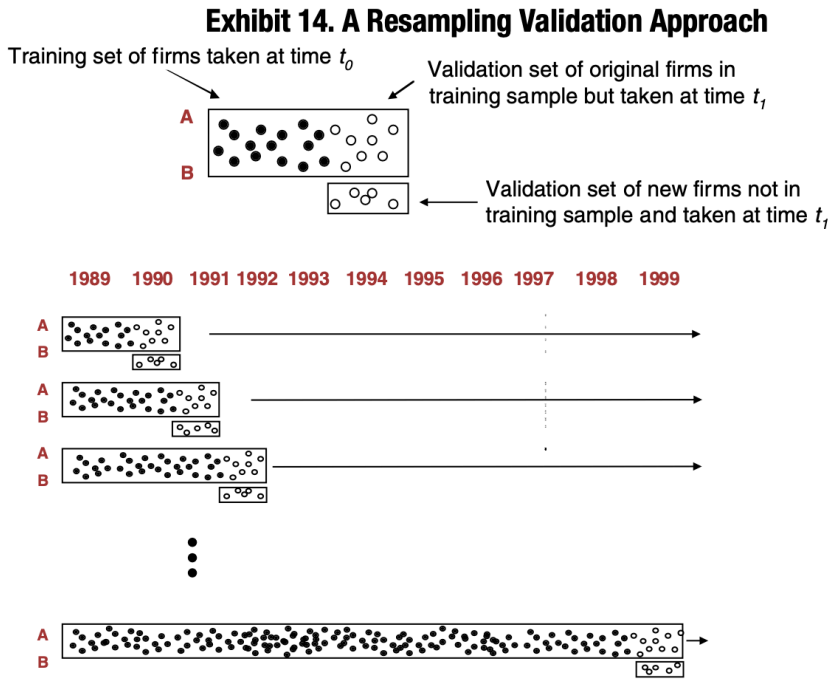


(a) Forecast for year t



(b) Forecast for year $t+1$

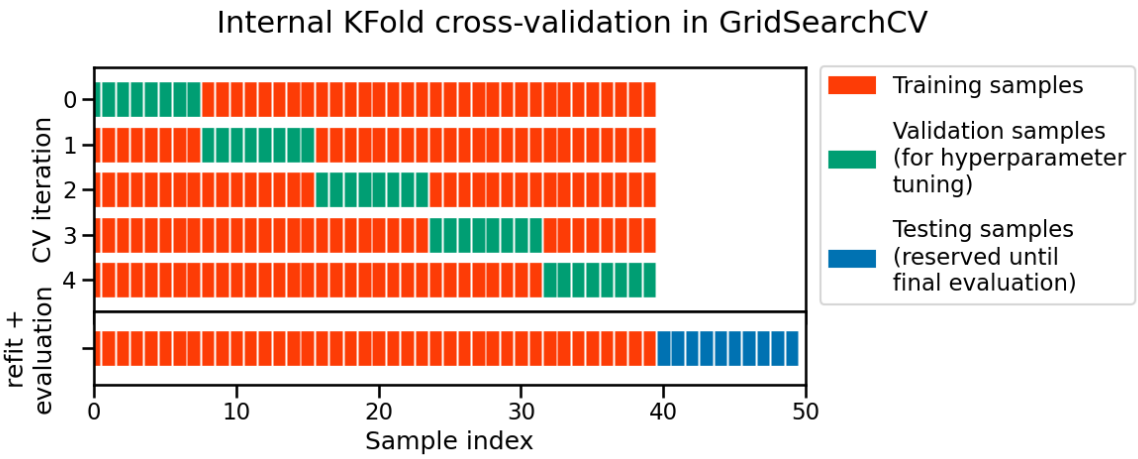
You might also implement something similar to the following scheme introduced in the Practical 1 and proposed by rating agency Moody's:

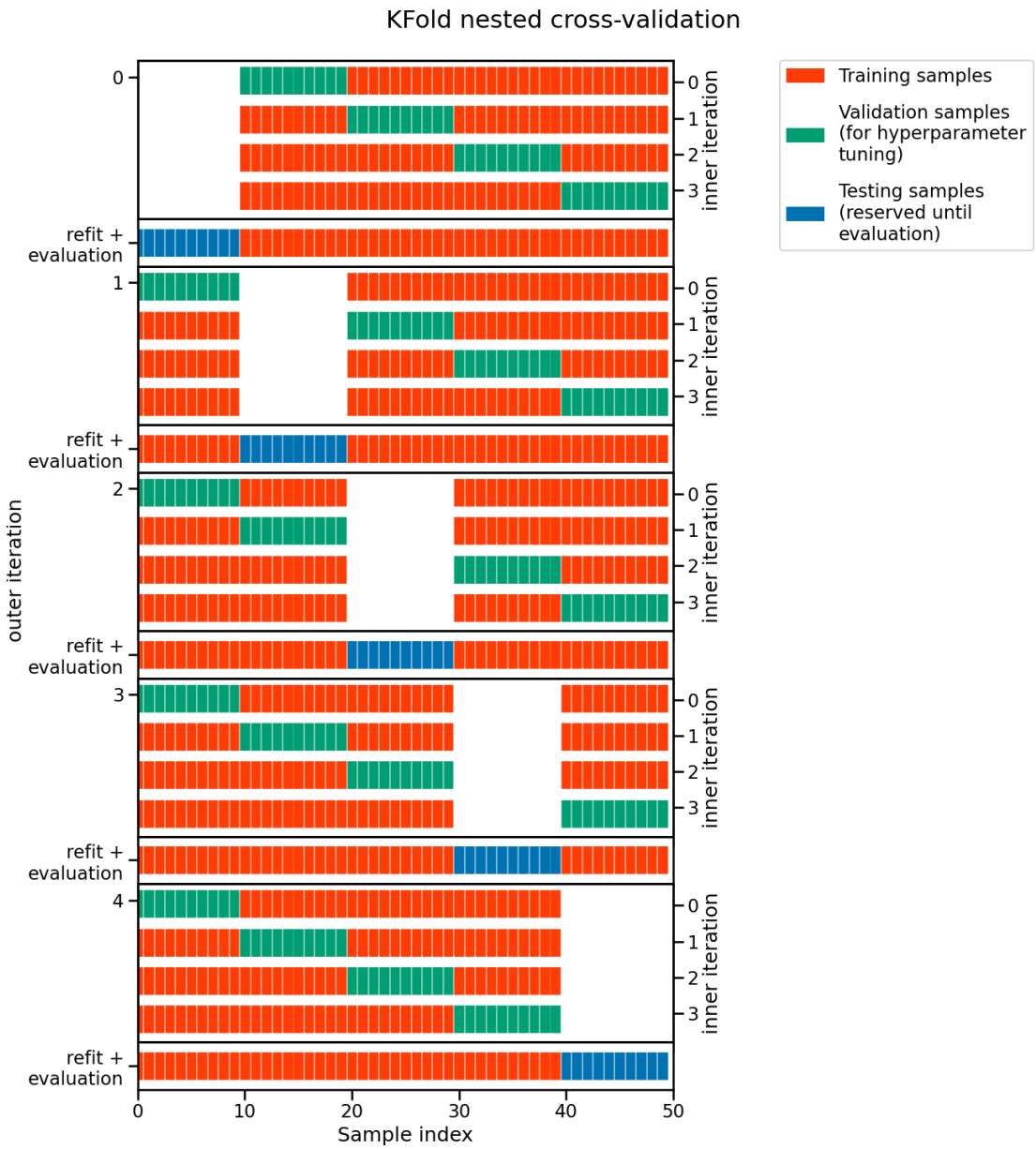


Moody's fits a model using a sample of historical data on firms and tests the model using both data on those firms one year later, and using data on new firms one year later (upper portion of exhibit). Dark circles represent in-sample data, used to estimate the model, and white circles represent testing data. We do "walk-forward testing" (bottom left) by fitting the parameters of a model using data through a particular year, and testing on data from the following year, and then inching the whole process forward one year.

Going further, at each step of the procedure you might perform a "inner Cross Validation", for example spliting multiple time the training set while holding out some companies (removing all the firm-years in training sample), to chose the best λ, on unseen companies.

Additional ideas and best practices about Cross validation (in particular for hyperparameters search) is available through `scikit-learn` documentation (here and here), see below for a generic comparison of K-fold:


Internal KFold cross-validation in GridSearchCV
CV iteration
0
1
2
3
4
refit + evaluation
0
10
20
30
40
50
Sample index
Training samples
Validation samples (for hyperparameter tuning)
Testing samples (reserved until final evaluation)



and nested K-fold:

4 Additional material

Below we remind some theoretical and practical elements about Logistic regression, model assessment in the context of statistical learning and variable selection methods.

We conclude with a case study showing the usage of Lasso and the more "controversial" stepwise methods on the Desbois data set.

Also, everything about Estimation / Interpretation / Inference has already been more thoroughly and rigorously covered in the theoretical part of the Scoring course and should be always used as a reference.

You don't have to read everything about this material, but Section 8 is of interest and has been presented in class. Also the case study in Section 9 shows how to "implement" Lasso with `R` package `glmnet` and goes deeper about the Cross Validation procedure in the context of Lasso penalty.

5 The Logistic Regression model

5.1 Informal introductory example

We provide here some intuitions leading to the Logistic Regression model using a simulated data set from James et al. (2021) (the **Default** data set).

This is a toy data set used for teaching purposes containing information on ten thousand customers.

The aim here is to assess which customers will ***default*** on their credit card debt (the target or response variable) based on the current credit card ***balance*** and other individual characteristics (the predictors or feature vector).

This is a binary classification problem as the ***default*** variable takes value in a discrete set (here binary). In the following we will denote Y the output or response variable and $X = (X_0, X_1, \cdots, X_{p-1})^T$ the feature vector or inputs.

The approach we follow is similar to (Hosmer et al., 2013) or (Cornillon et al., 2019). Both books use a similar example: the presence or absence of a Coronary Heart Disease (CHD) is explained with the age of an individual (the data set *chdage* is available in companion package *aplore3*).

We can start to explore the Default data with a scatterplot (Figure 1) of the target variable (***default***) with respect to a predictor (***balance***):

```
# Default data set (simulated) from ESLII/ISLR
default_data <- ISLR2::Default %>%
    as_tibble()

ggplot(default_data, aes(x=balance, y=default)) +
geom_point(alpha=0.2)
```

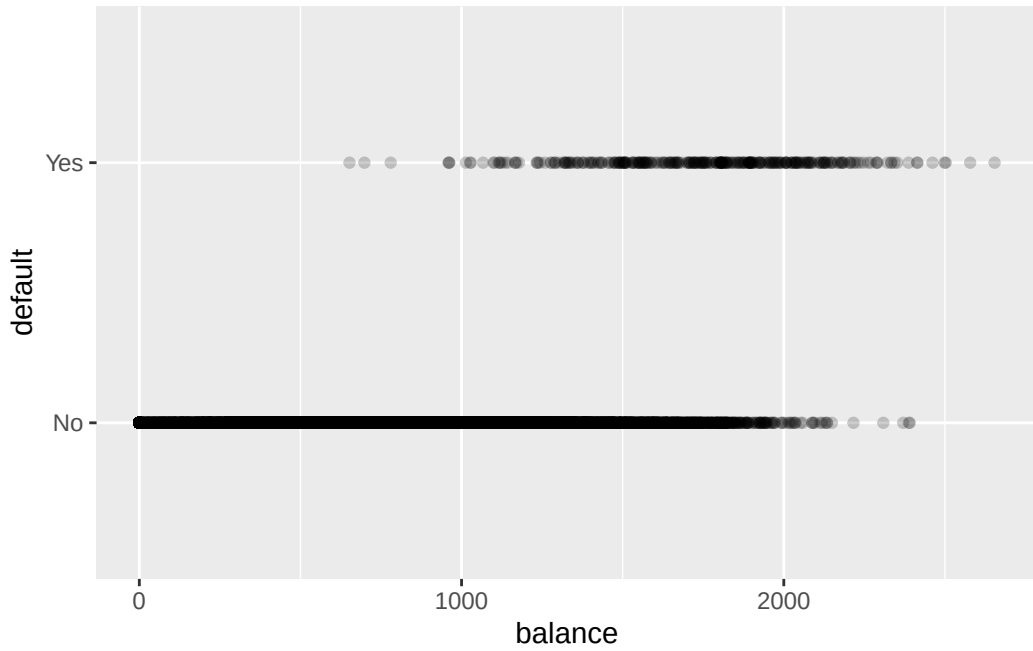
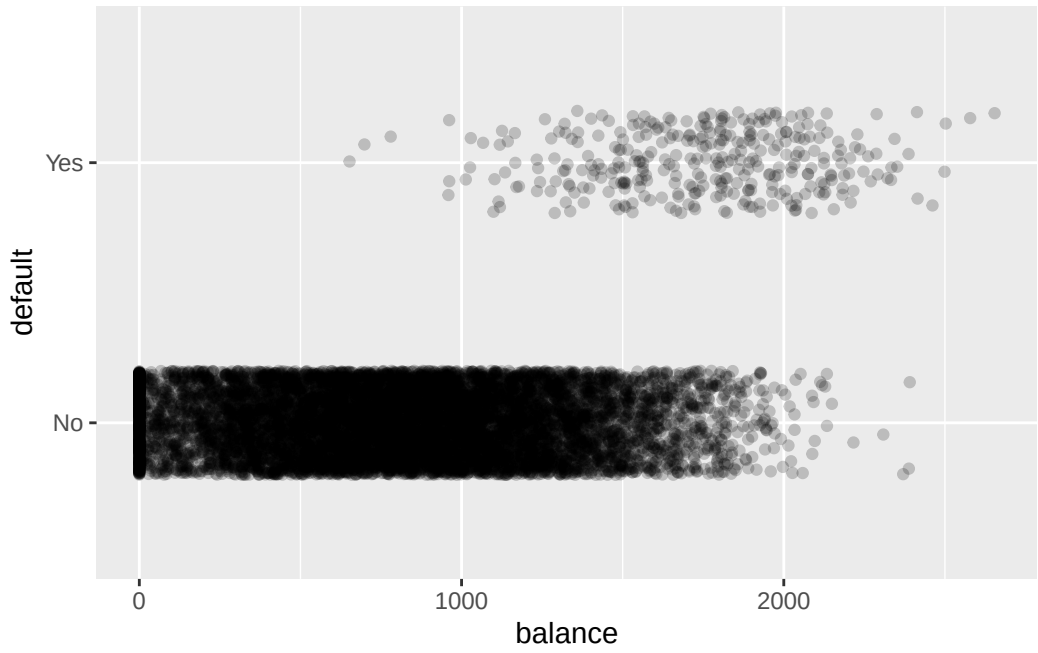


Figure 1: Scatterplot of variable ***default*** with respect to credit card ***balance*** for 10000 customers

In this scatterplot, all points fall on one of two parallel lines representing the absence (No) or occurrence (Yes) of ***default***. We "jitter" the data vertically to avoid overplotting. The plot below shows that the response variable is imbalanced towards the absence of default:

```
ggplot(default_data, aes(x=balance, y=default)) +
geom_jitter(alpha=0.2, height=0.2)
```



We also show the boxplots of credit cards ***balance*** with respect to ***default*** status:

```
ggplot(default_data,
       aes(x=default, y=balance)) +
geom_boxplot()
```

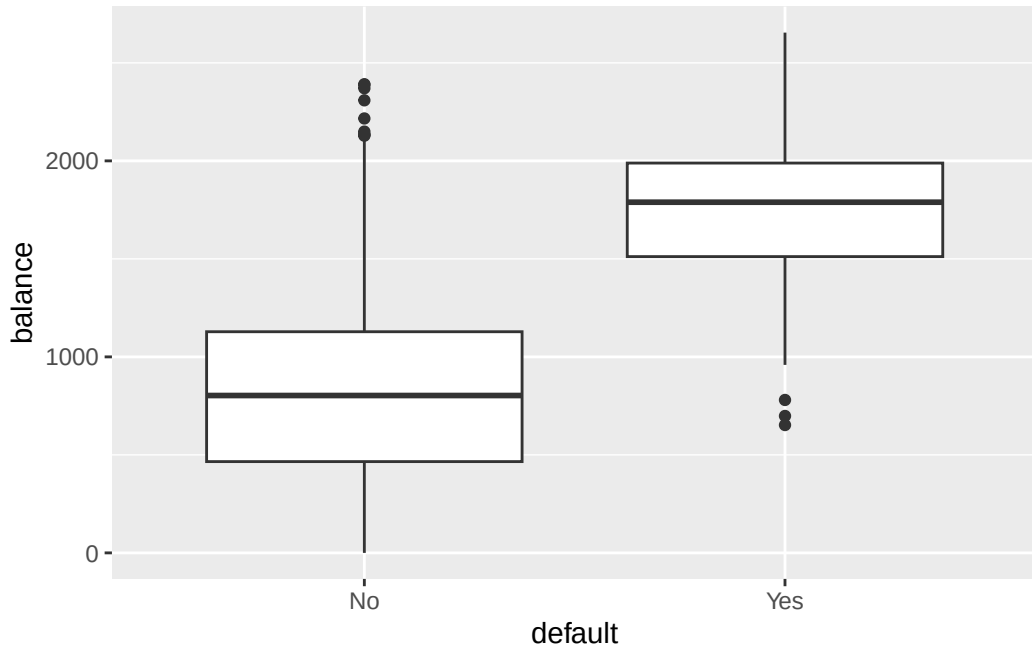


Figure 2: Variable ***balance*** with respect to ***default*** status

We can see from Figure 1 and Figure 2 that default tends to be more prevalent for accounts with a high balance. However it is difficult to guess a simple relationship between default and balance.

To investigate further we discretise the balance variables by classes of width 300$ and compute the mean of response variable (***default*** is Yes) within each balance class:

```
class_width <- 300
(default_data_binned <- default_data %>%
    mutate(balance_bins = cut(balance, breaks = seq(0, 3000, class_width),
                              right = FALSE, dig.lab = 4),
           min = floor(balance / class_width) * class_width,
           max = if_else(balance == 0 , 1,
                         # customers with 0$ balance should be long to [0, width) class
                         # or be excluded
                         ceiling(balance / class_width))  * class_width) %>%
    group_by(balance_bins, min, max, default) %>%
    summarize(n=n()) %>%
    ungroup() %>%
    pivot_wider(names_from = default, values_from = n) %>%
    replace_na(list(Yes = 0, No = 0)) %>%
```

```
    mutate(`Mean(default)` = round(Yes / (Yes + No), 4)))
```

```
# A tibble: 9 x 6
  balance_bins   min   max    No   Yes `Mean(default)`
  <fct>        <dbl> <dbl> <int> <int>           <dbl>
1 [0,300)          0   300  1497     0          0
2 [300,600)      300   600  1784     0          0
3 [600,900)      600   900  2305     3          0.0013
4 [900,1200)     900  1200  2098    19          0.009
5 [1200,1500)   1200  1500  1330    53          0.0383
6 [1500,1800)   1500  1800   527    96          0.154
7 [1800,2100)   1800  2100   115   114          0.498
8 [2100,2400)   2100  2400    11    41          0.788
9 [2400,2700)   2400  2700     0     7          1
```

In the following we map, by convention and for better readability, the response variable $Y \in \{Yes, No\}$ to $\{0, 1\}$:

$$Y = \begin{cases} 1 & \text{if customer defaulted on its credit card (ie default=Yes)} \\ 0 & \text{otherwise.} \end{cases}$$

Then we plot the mean of default (in red) within each balance class (of width 300$):

```
(default_occurrence <-
 ggplot(default_data %>%
          mutate(default = if_else(default == "Yes", 1, 0)),
        aes(x=balance, y=default)) +
 geom_point(alpha=0.2) +
 geom_segment(data = default_data_binned,
              aes(x = min, xend = max, y = `Mean(default)`, yend = `Mean(default)`),
              color = 'coral',
              linewidth=1.25))
```

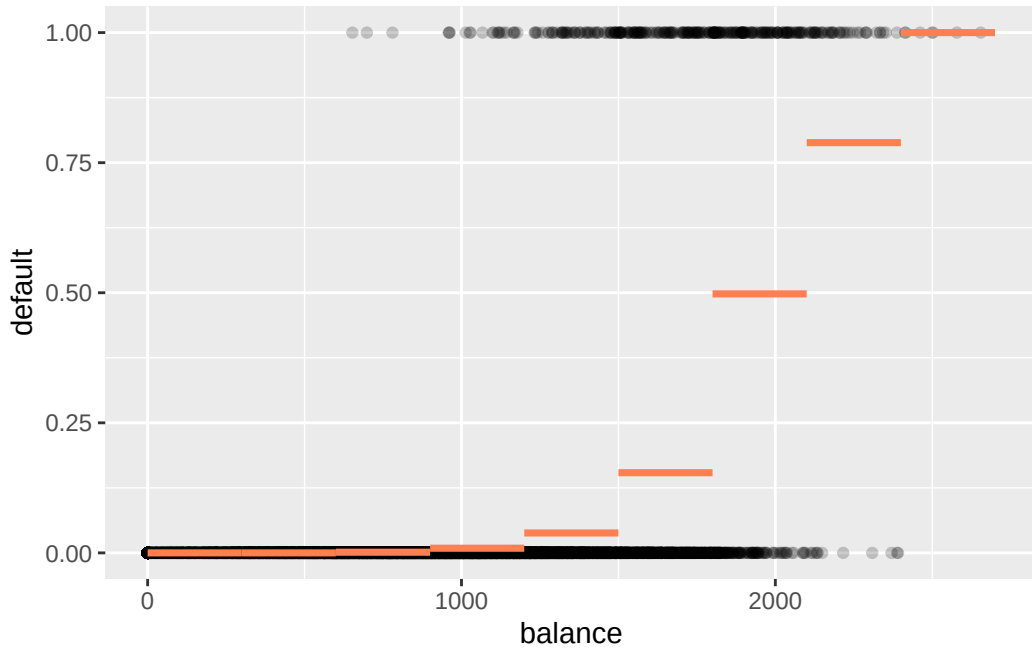


Figure 3: Mean occurrence of ***default*** within ***balance*** classes

The relationship between the mean occurrence of ***default*** and ***balance*** is easier to read.

Figure 3 clearly shows that as balance increases, the proportion of customers defaulting on their credit card increases.

We also notice that the mean default occurrence with respect to balance classes follows a kind of "S"-shaped curve or **sigmoid** function. Note that the shape depends on classes width and might change.

Going further and informally, considering that the mean of default occurrence is an estimate of $\mathbf{E}[Y|X = x]$ for each balance classes an idea would be to model:

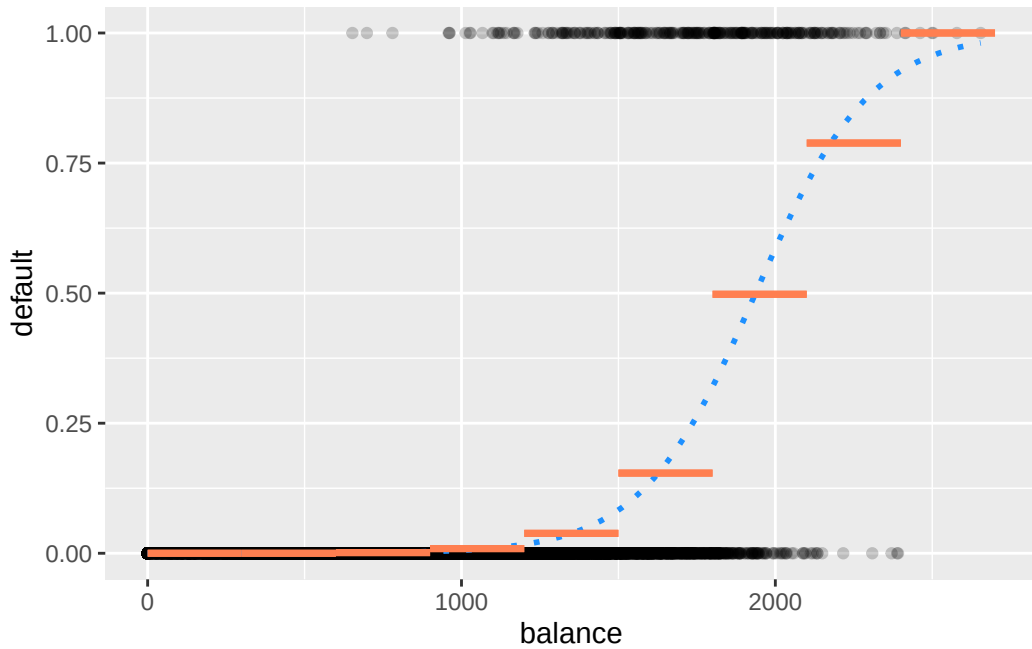
$$\mathbb{E}[Y|X = x] = \mu_\beta(x)$$

where μ_β is a **sigmoid** function in $[0, 1]$.

The Logistic Regression model uses the **sigmoid** function $\sigma : x \to \sigma(x) = \frac{e^x}{1+e^x}$ also known as logistic function.

```
(default_occurrence +
 geom_smooth(method = "glm",
             formula = y ~ x,
```

```
              method.args = list(family = "binomial"),
              se = FALSE,
              col = "dodgerblue",
              linetype = "dotted") +
  geom_segment(data = default_data_binned,
               aes(x = min, xend = max, y = `Mean(default)`, yend = `Mean(default)`),
               color = 'coral',
               linewidth=1.25))
```



Another approach would have been to treat Y as a quantitative response variable and fit a simple linear model:

```
linear <- lm(default ~ balance,
             data = default_data %>%
               mutate(default = if_else(default == "Yes", 1, 0)))
#summary(linear)
coeff_lm <- linear$coefficients
alpha <- coeff_lm["(Intercept)"]
beta <- coeff_lm["balance"]
```

```r
(ggplot(default_data %>%
          mutate(default = if_else(default == "Yes", 1, 0)),
        aes(x=balance, y=default)) +
 geom_point(alpha=0.2) +
 geom_segment(data = default_data_binned,
              aes(x = min, xend = max, y = `Mean(default)`, yend = `Mean(default)`),
              color = 'coral',
              size=1.25)+
 geom_line(data = tibble(x = seq(0,3000, 300)) %>%
               mutate(y = alpha + beta * x),
          aes(x = x, y = y),
          color = 'darkolivegreen',
          linetype = 'dotted'))
 # Alternatively we could have used the geom_smooth command
 # geom_smooth(method = "lm",
 #             formula = y ~ x,
 #             se = FALSE,
 #             col = "darkolivegreen",
 #             linetype = "dotted")
```

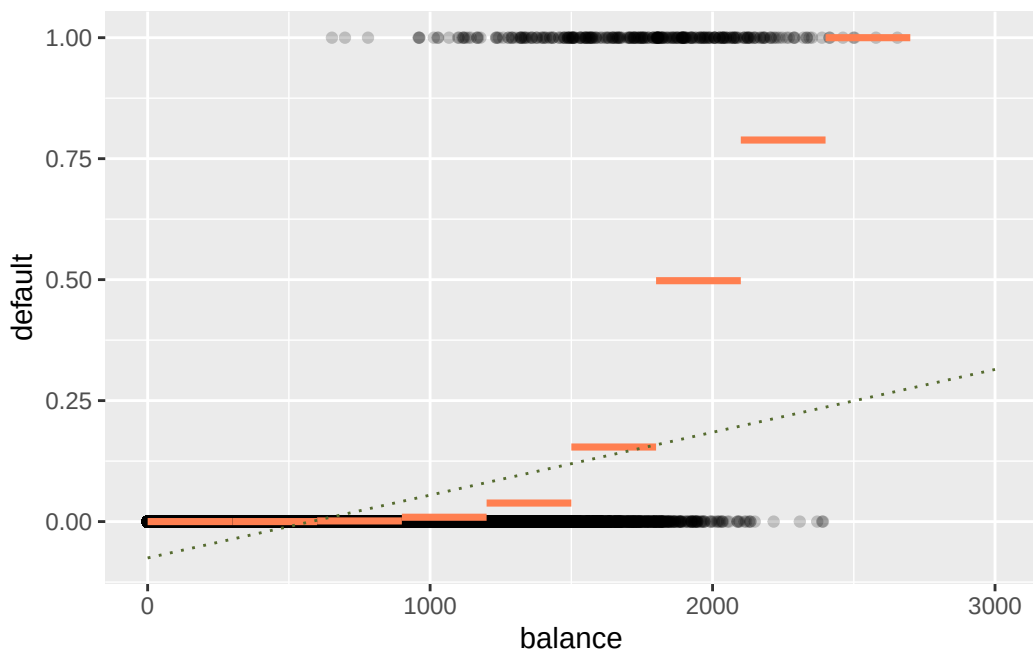


Figure 4: Fitting a linear model, ***default*** is treated as a quantitative response variable

Figure 4 shows that a linear model fails to fit the conditional probabilities. In particular, for low credit card balances the linear model shows a "negative probability of default". This is quite prominent here as response variable is imbalanced towards the No default category. For the same reasons, the least square method fails to correctly fit the category of interest (less than 0.25 probability).

Usually in such presentation (for example Hosmer et al. (2013)) data is more balanced and a linear model approximately fits the two classes. However in any case a linear model cannot confine the predicted value to $[0, 1]$ for all observations of predictors. Note that in the context of Scoring (ie High probability of default / High value of Scoring function), the linear model is nonetheless a candidate for a Scoring function (ie High/Low probability of default - High/Low value of Scoring function).

5.2 A more formal definition

We use the concepts that we have defined in the first part of the lesson (see slides 0_presentation_SE.pdf, section Scoring). The problem we are facing trying to predict the output default ($Y \in (0, 1)$) using a training set of inputs or feature vector X is a binary classification problem.

We remind that to estimate an optimal classifier for output $Y \in \{0, 1\}$ using input $X = (X_1, \cdots, X_p)$ one approach was to:

- model the joint distribution (or generative distribution) between X and Y,
- estimate $\eta(x)$ with:

$$\eta(x) = \mathbb{P}[Y = 1|X = x)] = \mathbb{E}[Y = 1|X = x)]$$

- $\eta(x)$ can be used as a Scoring function (ROC curve, choice of cutoff s)
- and we can use the classifier (usually $s = \frac{1}{2}$) to predict output Y:

$$f_s(x) = \begin{cases} 1 & \text{if } \eta(x) \geq s \\ 0 & \text{otherwise} \end{cases}$$

In the case of **Logistic Regression** classifier, we model:

$$Y|X = x \ \sim B(\eta(x))$$

with

$$\eta(x) = \sigma(x^T\beta) = \frac{exp(x^T\beta)}{1 + exp(x^T\beta)}$$

for some parameter $\beta = (\beta_1, \cdots, \beta_p) \in \mathbb{R}^p$, usually $x_1 = 1$ and β_1 is an intercept. σ is the sigmoid logistic function we have seen before.

In the literature is usual to denote $\eta(X) = p_\beta(X)$ or $\eta(X) = \pi_\beta(X)$.

From now, we will use the notation $p_\beta(X)$.

Defining logit : $x \to \log\left(\frac{x}{1-x}\right)$ we have:

$$\text{logit}(p_\beta(X)) = X^T\beta$$

The **Logistic Regression** model:

We are given $(x_i, y_i) \in \mathbb{R}^p \times \{0, 1\}$, $i = 1, \cdots, n$
The Logistic Regression model assumes that outputs y_i are independent Bernoulli with parameter $p_\beta(x_i)$ depending on x_i:

$$\text{logit}(p_\beta(x_i)) = x_i^T\beta$$

The Logistic Regression model defined above is a special case of more general family of models, the so-called Generalized Linear Models (GLM).

The family of GLM extends the applicability of linear-model ideas to data where responses are binary (e.g. Logistic Regression) or counts (e.g. Poisson Regression), among further possibilities. The concept emerged with Nelder and Wedderburn and has been studied extensively (see Nelder & Wedderburn (1972) or Cornillon et al. (2019)) .

The syntax to fit the Logistic model in R using `glm()` is:

$$\texttt{glm}(\text{y} \sim \text{x}, \texttt{data} = \text{dataframe}, \texttt{family} = \texttt{binomial}(\texttt{link} = \texttt{"logit"})$$

The formula y $\sim$ x depicts the model (i.e. inputs are X, output is Y) and the `data=` argument points to the training set contained in a R dataframe (or tibble). This is quite similar to the `lm()` function.

We also need to specify the distribution for the conditional Y values (binomial) and the link function (logit) via the `family=` argument.

For our example:

```r
glm_default <- glm(default ~ 1 + student + balance + income,
                   data = default_data,
                   family = binomial(link = "logit"))

glm_default <- glm(default ~ .,
                   data = default_data,
                   family = "binomial") # by default: link = "logit"
```

The command `summary` produces result summaries of the fitted model:

```r
summary(glm_default)
```

```
Call:
glm(formula = default ~ ., family = "binomial", data = default_data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
income       3.033e-06  8.203e-06   0.370  0.71152
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1571.5  on 9996  degrees of freedom
AIC: 1579.5

Number of Fisher Scoring iterations: 8
```

We will see later how to interpret or understand what is printed

A coefficient-wise output of the model can be obtained as a `tibble` using `tidy()` from package `broom`:

```r
broom::tidy(glm_default)
```

```
# A tibble: 4 x 5
  term            estimate  std.error statistic   p.value
```

```
  <chr>              <dbl>      <dbl>     <dbl>     <dbl>
1 (Intercept) -10.9        0.492        -22.1   4.91e-108
2 studentYes   -0.647      0.236         -2.74  6.19e-  3
3 balance       0.00574    0.000232      24.7   4.22e-135
4 income        0.00000303 0.00000820     0.370 7.12e-  1
```

Based on this output, the fitted model is (we have re-scaled balance and income for better readability):

$$\log\left(\frac{p_\beta(x_i)}{1-p_\beta(x_i)}\right) = -1.08 - 0.65(\mathbb{1}_{\text{student}_i=\text{Yes}}) + 5.74(\frac{\text{balance}_i}{1000}) + 0.03(\frac{\text{income}_i}{10000})$$

5.3 Estimation

5.3.1 Maximum Likelihood Estimation

We are given $(x_i, y_i) \in \mathbb{R}^p \times \{0,1\}$, $i = 1, \cdots, n$ where outputs y_i are independent Bernoulli with parameter $p_\beta(x_i)$ depending on x_i:

$$\text{logit}(p_\beta(x_i)) = x_i^T \beta$$

The parameters β of the Logistic Regression model are usually determined using Maximum Likelihood Estimation (MLE). It consists on finding β for which the joint probability of the observed data is greatest.

As y_i are independent the likelihood function (joint probability) is the product of the probability mass functions:

$$L(Y, \beta) = \prod_{i=1}^{n} p_\beta(x_i)^{y_i}(1 - p_\beta(x_i))^{1-y_i}$$

with $Y = (y_1, \cdots, y_n)$ and $\beta = (\beta_1, \cdots, \beta_p)$.

We seek to maximize the likelihood function over β, it is equivalent but easier to maximize the log-likelihood:

$$
\begin{aligned}
\ell(Y,\beta) = \log L(Y,\beta) &= \sum_{i=1}^{n} \left(y_i \log(p_\beta(x_i)) + (1-y_i)\log(1-p_\beta(x_i)) \right) \\
&= \sum_{i=1}^{n} \left(y_i \log(\frac{p_\beta(x_i)}{1-p_\beta(x_i)}) + \log(1-p_\beta(x_i)) \right) \\
&= \sum_{i=1}^{n} \left(y_i x_i^T \beta - \log(1+\exp(x_i^T\beta) \right)
\end{aligned}
$$

If the MLE $\hat{\beta}$ exists, the gradient of log-likelihood satisfies (first order necessary condition):

$$
\nabla \ell(Y,\beta) = \left(\frac{\partial \ell(Y,\beta)}{\partial \beta_1}, \cdots, \frac{\partial \ell(Y,\beta)}{\partial \beta_p} \right) = \mathbf{0}
$$

We have for $j = 1, \cdots, p$:

$$
\frac{\partial \ell(Y,\beta)}{\partial \beta_j} = \sum_{i=1}^{n} \left(y_i x_{ij} - x_{ij} \frac{\exp(x_i^T\beta)}{1+\exp(x_i^T\beta)} \right) = \sum_{i=1}^{n} x_{ij} \left(y_i - p_\beta(x_i) \right)
$$

In vector form:

$$
\nabla \ell(Y,\beta) = \sum_{i=1}^{n} x_i \left(y_i - p_\beta(x_i) \right) = X^T(Y - P_\beta)
$$

where:

$$
X = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix} = \begin{pmatrix} x_1^T \\ x_2^T \\ \vdots \\ x_n^T \end{pmatrix} \in \mathbb{R}^{n\times(p)}, \quad Y = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} \quad \textit{and} \quad P_\beta = \begin{pmatrix} p_\beta(x_1) \\ p_\beta(x_2) \\ \vdots \\ p_\beta(x_n) \end{pmatrix}
$$

In the literature $\nabla \ell(Y,\beta)$ is denoted as the Fisher's score function $S(\beta)$, if the MLE $\hat{\beta}$ exists, we have:

$$
S(\hat{\beta}) = \nabla \ell(Y,\hat{\beta}) = X^T(Y - P_{\hat{\beta}}) = 0
$$

Solving this equation involves solving p non-linear equations in β:

$$
y_1 x_{1j} + \cdots + y_n x_{nj} = x_{1j} \frac{\exp(x_1^T\beta)}{1+\exp(x_1^T\beta)} + \cdots + x_{nj} \frac{\exp(x_n^T\beta)}{1+\exp(x_n^T\beta)}, \quad j = 1, \cdots, p
$$

5.3.2 Numerical methods

In practice we use numerical methods to solve these non-linear equations as no closed-form solution exist.

If we assume that $rank(X) = p$, we will have that $S(\beta)$ is concave in β hence if we find a local maximum it is a global maximum.

We have for $(k, l) \in (1, \cdots, p)^2$:

$$\begin{aligned}\frac{\partial \ell}{\partial \beta_k \partial \beta_l}(\beta) =& \frac{\partial}{\partial \beta_k} \sum_{i=1}^{n} x_{il}(y_i - \frac{\exp(x_i^T \beta)}{1 + \exp(x_i^T \beta)}) \\ =& - \sum_{i=1}^{n} x_{il} x_{ik} \frac{\exp(x_i^T \beta)}{(1 + \exp(x_i^T \beta))^2} \\ =& - \sum_{i=1}^{n} x_{ik} p_\beta(x_i)(1 - p_\beta(x_i)) x_{il}\end{aligned}$$

We obtain that in matrix form:

$$H(\beta) = \nabla^2 \ell(Y, \beta) = -X^T W_\beta X$$

where:

$$W_\beta = \begin{pmatrix} p_\beta(x_1)(1 - p_\beta(x_1)) & \cdots & \cdots \\ \vdots & \ddots & \vdots \\ \cdots & \cdots & p_\beta(x_n)(1 - p_\beta(x_n)) \end{pmatrix}$$

We have $p_\beta(x_i)(1 - p_\beta(x_i)) \geq 0$ hence $W(\beta)$ is semi-definite negative and since $rank(X) = p$, $H(\beta)$ is concave.

It is shown in (Albert & Aanderson, 1984) that if additionally there is no complete separation in the training set:

then the MLE exists and is unique.

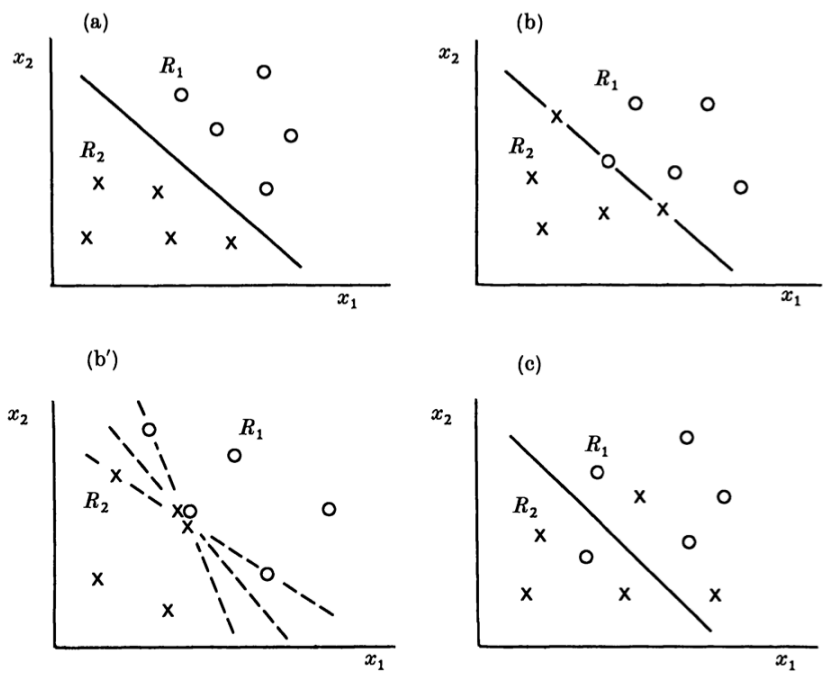


Fig. 1. Possible configurations of sample points in the case of two variables, x_1 and x_2, and two groups, H_1, shown by circles, and H_2, shown by crosses. Regions R_1 and R_2 define corresponding allocation rule. (a) Complete separation. (b) Quasicomplete separation $\rho(X^q) = 2$. (b′) Quasicomplete separation $\rho(X^q) = 1$; at the point of intersection of the lines, there are three observations, one from H_1, and two from H_2. (c) Overlap.

5.3.2.1 The Newton-Raphson (ie Fisher Scoring) method

In practice the Newton-Raphson method is used to solve the equation:

$$S(\beta) = \nabla \ell(Y, \beta) = X^T(Y - P_\beta) = 0$$

Using Taylor expansion of Score $S(\beta)$:

$$S(\hat{\beta}) \approx S(\beta^{(k)}) + H(\beta^{(k)})(\hat{\beta} - \beta^{(k)})$$

and starting from an initial guess of $\beta = \beta_0$, the Newton-Raphson update formula is:

$$\beta^{(k+1)} = \beta^{(k)} - H^{-1}(\beta^{(k)})S(\beta^{(k)})$$

We show below a naive implementation of Newton-Raphson method to estimate β (also known as Fisher Scoring algorithm in the context of Logistic Regression)

```
# We put the data frame in matrix form
# also adding an intercept
X <- cbind(rep(1, nrow(default_data)),
                          as.matrix(default_data %>% select(balance, income)))
colnames(X) <-  c("(Intercept)", "balance", "income")
n <- nrow(X)

# We extract the output as vector
Y <- default_data %>% mutate(default = if_else(default=='Yes', 1, 0)) %>% pull(default)


# We set an initial guess for beta and criterion for stopping
beta <- c(0.01, 0.0, 0.0)
nb_iter <- 25
tol <- 1e-4

lr_solve <- function(X, Y, beta, nb_iter, tol){
    for(i in 1:nb_iter){
        # first compute p_beta(X)
        p_beta <- exp(X %*% beta) / (1 + exp(X %*% beta))

        # then the Score
        Score_beta <- t(X) %*% (Y-p_beta)

        # and the Hessian
```

```r
        W_beta <- matrix(0, n, n)
        diag(W_beta) <- p_beta*(1-p_beta)

        Hessian_beta <- -t(X) %*% W_beta %*% X

        # we update beta
        new_beta <- beta - solve(Hessian_beta) %*% Score_beta

        # we check for convergence
        if(t(beta-new_beta) %*% (beta-new_beta) < tol){
            return(list(beta = beta, hessian = Hessian_beta, nb_iter = i))
        }
        beta <- new_beta
    }
    return(list(beta = beta, hessian = Hessian_beta, nb_iter = i))
}

sol <- lr_solve(X, Y, beta, nb_iter, tol)
```

We verify that R `glm()` and our algorithm give the same coefficients:

- Newton-Raphson:

```r
print(round(t(sol$beta),6))
```

```
     (Intercept)  balance  income
[1,]   -11.53791 0.005646 2.1e-05
```

- R `glm()`:

```r
glm_bal_inc <- glm(default ~ balance + income,
                   data = default_data,
                   family = "binomial")

print(round(coef(glm_bal_inc ),6))
```

```
(Intercept)     balance      income
 -11.540468    0.005647    0.000021
```

In the next sections (Interpretation, Confidence intervals, Tests) we will try to understand the outputs of the `glm()` function from a statistical viewpoint.

5.3.2.2 The Iterative Reweighted Least Square (IRLS) method

There is an equivalent approach to the the Newton-Raphson described in the literature as Iterative Reweighted Least Square (IRLS).

The Newton-Raphson update formula rewrites:

$$\begin{aligned}
\beta^{(k+1)} &= \beta^{(k)} - H^{-1}(\beta^{(k)})S(\beta^{(k)}) \\
&= \beta^{(k)} + (X^T W_{\beta^{(k)}} X)^{-1} X^T (Y - P_{\beta^{(k)}}) \\
&= (X^T W_{\beta^{(k)}} X)^{-1} (X^T W_{\beta^{(k)}} X)\beta^{(k)} + (X^T W_{\beta^{(k)}} X)^{-1} X^T (Y - P_{\beta^{(k)}}) \\
&= (X^T W_{\beta^{(k)}} X)^{-1} X^T W_{\beta^{(k)}} \left(X\beta^{(k)} + W_{\beta^{(k)}}^{-1} (Y - P_{\beta^{(k)}}) \right) \\
&= (X^T W_{\beta^{(k)}} X)^{-1} X^T W_{\beta^{(k)}} Z_{\beta^{(k)}}
\end{aligned}$$

where:

$$Z_{\beta^{(k)}} = X\beta^{(k)} + W_{\beta^{(k)}}^{-1} (Y - P_{\beta^{(k)}})$$

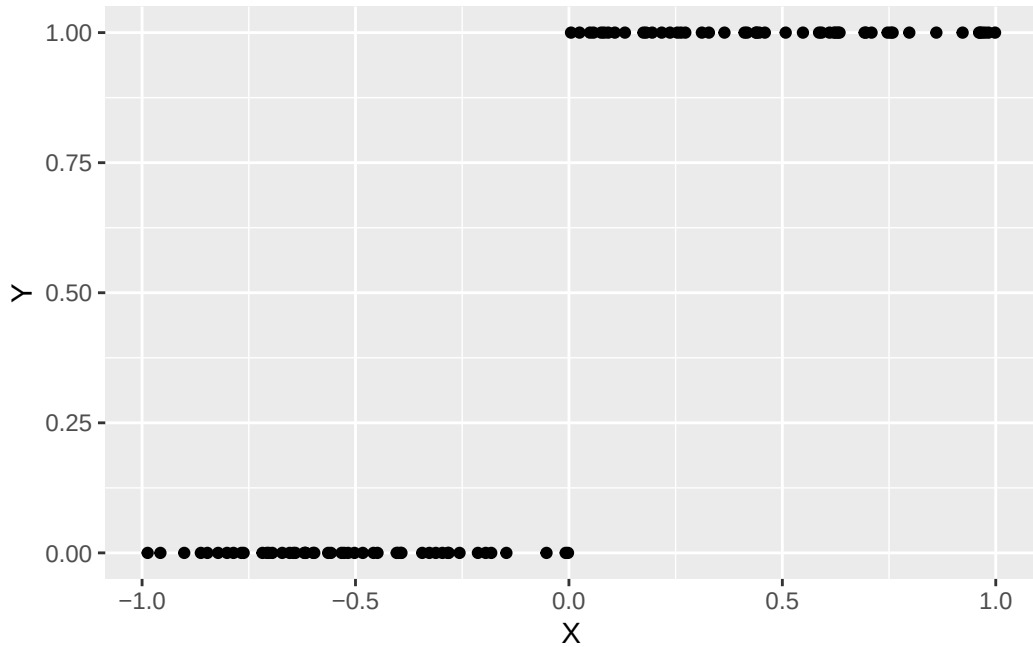
$\beta^{(k+1)} = (X^T W_{\beta^{(k)}} X)^{-1} X^T W_{\beta^{(k)}} Z_{\beta(k)}$ corresponds to the solution of a weighted $(W_{\beta^{(k)}})$ linear regression of $Z_{\beta^{(k)}}$ by X. As an exercise you can implement this algorithm.

5.3.2.3 Example where MLE is not finite

To conclude on the numerical aspects, we signal a special and extreme case where the iterative algorithm won't converge. The theoretical aspect is covered in (Albert & Aanderson, 1984).

We simulate a perfectly separated data set. Here $X \in [-1, 1]$ and $Y \in \{0, 1\}$:

```
set.seed(1987)
X <- c(runif(n = 50, min = -1, max = 0),
       runif(n = 50, min = 0, max = 1))
Y <- c(rep(0, 50), rep(1, 50))

tbl_separated <- tibble(X,Y)
ggplot(tbl_separated) + geom_point(aes(X, Y))
```

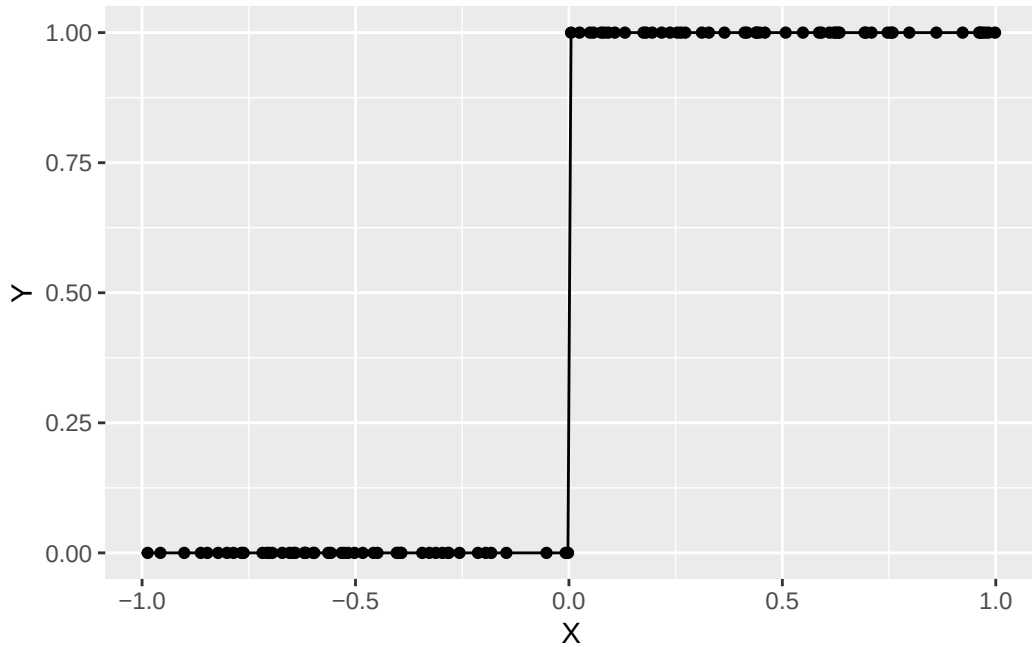


In this setting the iterative algorithm fails to converge and coefficient "saturates" to a high/low value (while it should go to infinite):

```
glm_separated <- glm(Y ~ X,
                     data = tbl_separated,
                     family="binomial")

glm_separated$coef
```

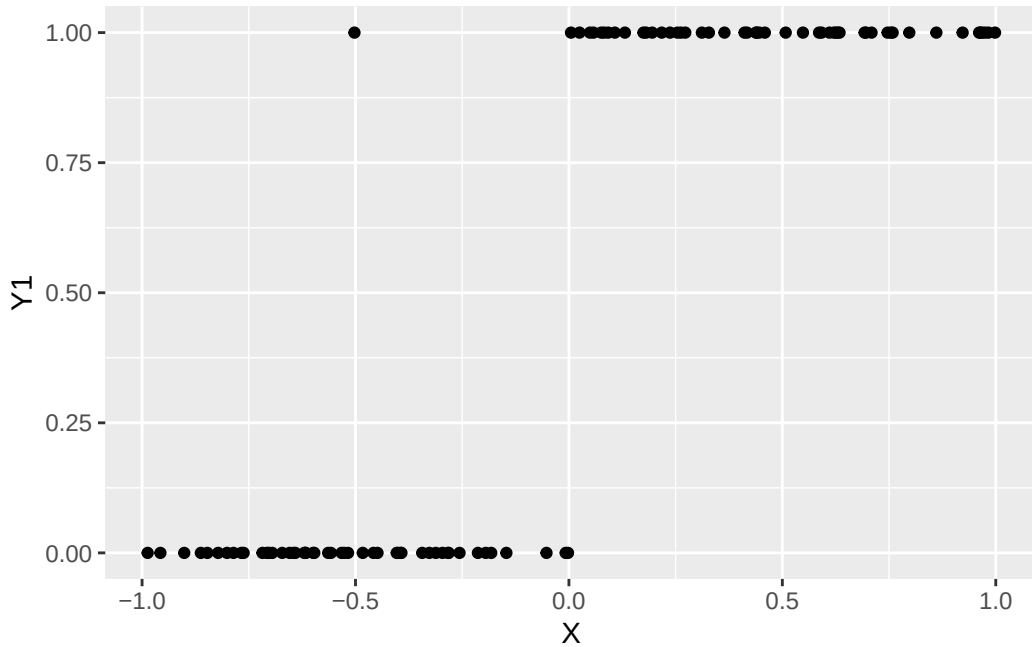
```
(Intercept)           X
  -5.763399 4227.385776
```

```
separated_fit <- broom::augment(glm_separated, type.predict = "response")
ggplot(separated_fit) +
    geom_point(aes(X, Y)) +
    geom_line(aes(X,.fitted))
```



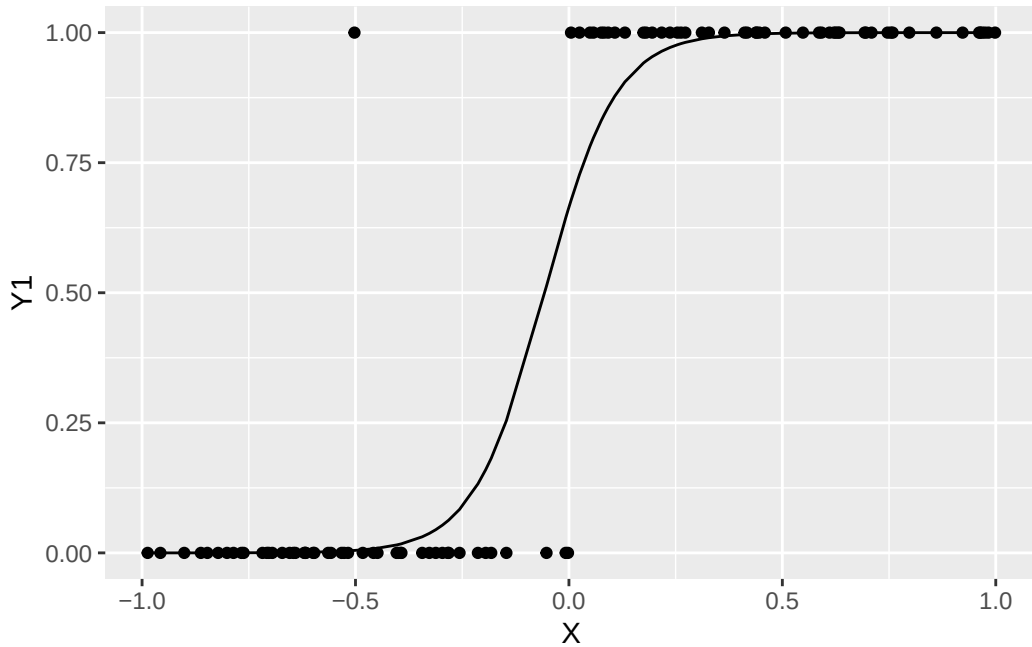
Now we slightly modify the data set, changing a y observation with $x \in [-1, 0]$ from 0 to 1.

```
Y1 <- Y
Y1[25] <- 1
tbl_overlap <- tibble(X,Y1)
ggplot(tbl_overlap) + geom_point(aes(X, Y1))
```



The iterative algorithm converges again and the impact on the Scoring function (i.e. $\mathbb{P}[Y = 1|X = x)$) and the decision rule (shifting left below $X = O$) is not negligible for a one point change:

```
glm_overlap <- glm(Y1 ~ X,
                   data = tbl_separated,
                   family="binomial")
overlap_fit <- broom::augment(glm_overlap, type.predict = "response")
ggplot(overlap_fit) +
    geom_point(aes(X, Y1)) +
    geom_line(aes(X,.fitted))
```



Nonetheless the case we described is very unlikely to happen in a real life setting, and a good data set exploration should avoid such trap. More details can be found in the document `Separation and Convergence Issues in Logistic Regression.pdf`

5.3.3 Logistic Regression as a machine learning approach

We have another look at the log-likelihood equation stated before, we have:

$$\begin{aligned}
\ell(Y,\beta) = \log L(Y,\beta) &= \sum_{i=1}^{n} \big(y_i \log(p_\beta(x_i)) + (1-y_i)\log(1-p_\beta(x_i))\big) \\
&= -\sum_{i=1}^{n} \ell_{logistic}\left(p_\beta(x_i), y_i\right) \\
&= -n\hat{\mathrm{R}}(p_\beta)
\end{aligned}$$

where $\ell_{logistic} : \{0,1\} \times \{0,1\} \to \mathbb{R}^+$:

$$\ell_{logistic}(y,z) = -y\log(z) - (1-y)\log(1-z) = \begin{cases} -\log(z) & \text{if } y = 1 \\ -\log(1-z) & \text{if } y = 0 \end{cases}$$

and $\hat{\mathrm{R}}(p_\beta)$ is the empirical risk on the training set.

Estimating β by maximizing the log-likelihood is equivalent to minimizing with respect to β the empirical risk of p_β for the logistic loss. Note that usually in the context of machine learning and logistic loss, the output Y is relabeled to $\{-1, 1\}$

6 Interpretation

```r
summary(glm_default)
```

```
Call:
glm(formula = default ~ ., family = "binomial", data = default_data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
income       3.033e-06  8.203e-06   0.370  0.71152
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1571.5  on 9996  degrees of freedom
AIC: 1579.5

Number of Fisher Scoring iterations: 8
```

Based on this output, the fitted model is:

$$\log\left(\frac{p_\beta(x_i)}{1-p_\beta(x_i)}\right) = -1.08 - 0.65(\mathbb{1}_{\text{student}_i=\text{Yes}}) + 0.57(\frac{\text{balance}_i}{100}) + 0.3(\frac{\text{income}_i}{1,000})$$

Please note that we have re-scaled balance (dividing by a factor 100) and income (dividing by a factor $10,000$) to more natural units for better readability, so that coefficients are also re-scaled (multiplied by factor $100/10,000$).

6.1 Coefficients

Tip

We cannot interpret the coefficients in the same manner as we interpret coefficients from a linear model, as the outcome is now expressed in "logits":

- The predicted logit (or as we will see later log-odds) of defaulting for non-students with zero balance and income are -1.08.
- Each one-unit difference in $\frac{\text{balance}}{100}$ is associated with a difference of 0.57 in the predicted logit of defaulting.
- Each one-unit difference in $\frac{\text{income}}{10{,}000}$ is associated with a difference of 0.03 in the predicted logit of defaulting.

6.2 Odds

We remind the following relationship:

$$\text{logit}(p_\beta(x)) = \log(\frac{p_\beta(x)}{1 - p_\beta(x)}) = x^T\beta$$

The ratio on which we take the logarithm is called odds:

$$odd_\beta(x) = \frac{p_\beta(x)}{1 - p_\beta(x)} = exp(x^T\beta)$$

It represents the chance an event occurs ($p_\beta(x)$) versus the chance that same event does not occur ($1 - p_\beta(x)$).

Odds are an alternative scale to probability for representing chance.

They arose as a way to express the payoffs for bets. An even bet means that the winner gets paid an equal amount to that staked.

A 3–1 against bet would pay \$3 for every \$1 bet, while a 3–1 on bet would pay only \$1 for every \$3 bet.

For an event A we have $odds(A) \in [0, +\infty[$, and $odds(A) > 1$ if $\mathbf{P}(A) > 0.5$.

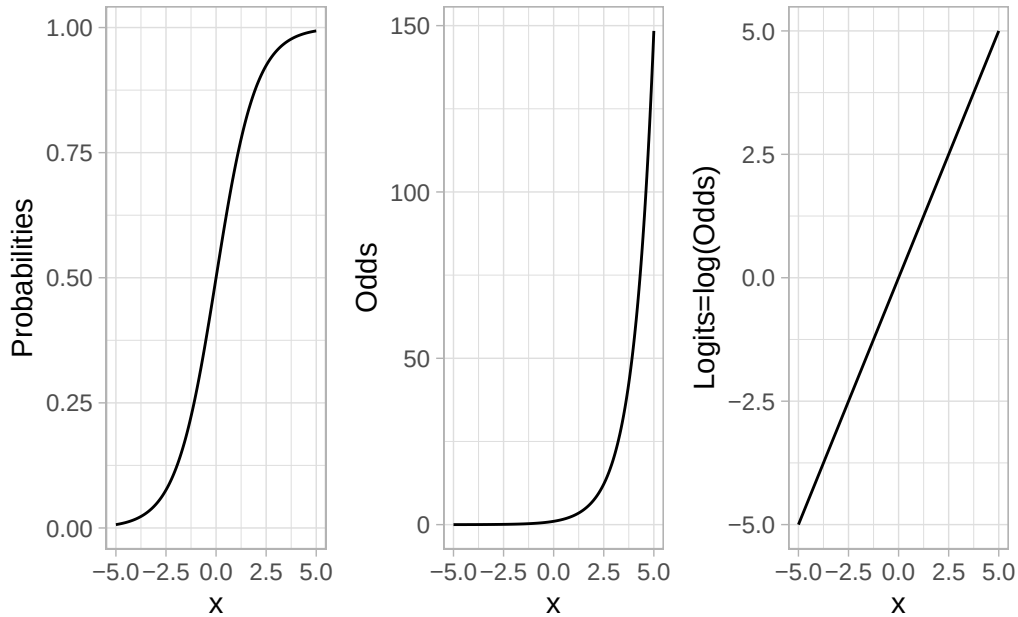
Example: $P(A) = \frac{3}{5}$ is equivalent to $Odds(A) = 1.5$. It means A happens 1.5 more often than its complement Not A.

We can also rewrite:

$$p_\beta(x) = \frac{odds_\beta(x)}{1 + odds_\beta(x)}$$

To set these ideas, for a variable x in $[-5, 5]$, we plot the logistic curve (i.e. the probabilities) together with the odds and the logits (ie log(odds)):

```
# Create w values and transformed values
data_logistic = tibble(x = seq(from = -5, to = 5, by = 0.01)) %>%
                  mutate(probs = exp(x) / (1 + exp(x)), # logistic curve
                         odds = probs / (1 - probs),
                         logits = log(odds))

# View data
# Logistic curve / sigmoid (probabilities)
p1 <-  ggplot(data = data_logistic , aes(x = x, y = probs)) +
  geom_line() +
  theme_light() +
  ylab("Probabilities")

# Exponential curve (odds)
p2 <-  ggplot(data = data_logistic , aes(x = x, y = odds)) +
  geom_line() +
  theme_light() +
  ylab("Odds")

# Linear curve (log-odds)
p3 <-  ggplot(data = data_logistic, aes(x = x, y = logits)) +
  geom_line() +
  theme_light() +
  ylab("Logits=log(Odds)")

library(patchwork)

p1 + p2 + p3
```



Tip

Interpret the coefficients in terms of odds:

- The coefficient of balance is $\beta_{balance} = 0.005737$. Hence an increase of balance by 100$ increases odds for default by a factor of $\exp(\beta_{balance} * 100) = 1.77$.
- The coefficient of income is $\beta_{income} = 3 \times 10^{-6}$. Hence an increase of income by 10000$ increases odds for default by a factor of $\exp(\beta_{income} * 100) = 1.03$.
- The coefficient of "being a student" is $\beta_{studentYes}$ = -0.647. Hence being a student decreases odds for default by a factor of $\exp(\beta_{studentYes}) = 0.52$.

6.3 Odds ratio

For two observations x and $\tilde{x}$ we define odds ratio as:

$$OR(x, \tilde{x}) = \frac{odds(x)}{odds(\tilde{x})}$$

Odds ratio are used to compare probabilities between two observations:

- $OR(x, \tilde{x}) = 1 \Leftrightarrow p(x) = p(\tilde{x})$
- $OR > 1 \Leftrightarrow p(x) > p(\tilde{x})$

- $OR < 1 \Leftrightarrow p(x) < p(\tilde{x})$

They are also used to measure the impact of a predictor:

$$OR(x, \tilde{x}) = \exp(\beta_1(x_1 - \tilde{x_1})) \cdots exp(\beta_p(x_p - \tilde{x_p}))$$

Choosing $(x, \tilde{x})$ differing by only one predictor x_j:

$$OR(x, \tilde{x}) = \exp(\beta_j(x_j - \tilde{x_j}))$$

In other words, $exp(\beta_j)$ is the odds ratio associated with a one-unit increase in the x_j.

More on odds ratio interpretation can be found here.

7 Inference

7.1 Asymptotic properties of MLE

It can be proven that under certain assumptions (see for example Gourieroux (1981) or Fahrmeir (1986)), the Maximum Likelihood Estimator has the following asymptotic properties:

$$\hat{\beta} \xrightarrow{p} \beta, \text{ as n} \to \infty$$

and

$$\sqrt{n}(\hat{\beta} - \beta) \xrightarrow{\mathcal{L}} \mathcal{N}(0, \mathcal{I}(\beta)^{-1}), \text{ as n} \to \infty$$

where:

$$\mathcal{I}(\beta) = -\mathbb{E}[\nabla^2 \ell(Y, \beta)] = -\frac{1}{n}\nabla^2 \ell(Y, \beta) = \frac{1}{n} X^T W_\beta X$$

where $\mathcal{I}(\beta)$ is the Fisher information matrix. In the case of Logistic Regression, Fisher information matrix equals the Observed information matrix.

The asymptotic property rewrites:

$$(\hat{\beta} - \beta)^T n \mathcal{I}(\beta)(\hat{\beta} - \beta) \xrightarrow{\mathcal{L}} \chi^2_p$$

As $\mathcal{J}(\beta)$ is unknown we use instead $\mathcal{J}(\hat{\beta}) = \frac{1}{n} X^T W_{\hat{\beta}} X$. Since $\hat{\beta} \xrightarrow{p} \beta$ and p_β continuous in β it can be shown that:

$$(\hat{\beta} - \beta)^T X^T W_{\hat{\beta}} X (\hat{\beta} - \beta) \xrightarrow{\mathcal{L}} \chi^2_p$$

Or equivalently:

$$\hat{\beta} - \beta \xrightarrow{\mathcal{L}} \mathcal{N}(0, (X^T W_{\hat{\beta}} X)^{-1})$$

7.2 Wald statistics

7.2.1 Confidence intervals

Using the preceding asymptotic properties we can derive confidence interval and tests for the coefficients β_j, $j = 1, \cdots, p$ of the model:

$$\frac{\hat{\beta}_j - \beta_j}{\hat{\sigma}_j} \xrightarrow{\mathcal{L}} \mathcal{N}(0, 1)$$

where $\hat{\sigma}_j^2 = s.e.(\hat{\beta}_j)^2$ denotes the $j - th$ term of $(X^T W_{\hat{\beta}} X)^{-1}$ diagonal.

The typical formula for a $1 - \alpha$ confidence interval is:

$$\hat{\beta}_j \pm z_{1-\alpha/2} \hat{\sigma}_j$$

where $z_{1-\alpha/2}$ is the $(1 - \alpha/2)$ quantile of the standard normal distribution.

Going further, the asymptotic properties of MLE also allow to test the "statistical significance" of each coefficient in the model, the Wald test.

Denoting: H_0: $\beta_j = 0$ and H_1: $\beta_j \neq 0$ we have under H_0:

$$\frac{\hat{\beta}_j}{\hat{\sigma}_j} \xrightarrow{\mathcal{L}} \mathcal{N}(0, 1)$$

We will reject H_0 at level α if the absolute of the observed value $\frac{\hat{\beta}_j}{\hat{\sigma}_j}$ (denoted in `glm` output as `z value`) is above the $(1 - \alpha/2)$ quantile of the standard normal distribution.

```r
glm_bal_inc <- glm(default ~ balance + income,
                   data = default_data,
                   family = "binomial")
summary(glm_bal_inc)
```

```
Call:
glm(formula = default ~ balance + income, family = "binomial",
    data = default_data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.154e+01  4.348e-01 -26.545  < 2e-16 ***
balance      5.647e-03  2.274e-04  24.836  < 2e-16 ***
income       2.081e-05  4.985e-06   4.174 2.99e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1579.0  on 9997  degrees of freedom
AIC: 1585

Number of Fisher Scoring iterations: 8
```

Said differently, we reject H_0 at level α when $p = \mathbf{P}(|z| > |\frac{\hat{\beta}_j}{\hat{\sigma}_j}|) < \alpha$.

p is called the p-value.

The output of `glm` in `R` shows:

- $\hat{\beta}_j$ as `Estimate`,
- $\hat{\sigma}_j$ as `Std. Error`,
- the absolute of the observed test statistic $\frac{\hat{\beta}_j}{\hat{\sigma}_j}$ as `z value`,
- and the p-value as `Pr(>|z|)`

We obtain the `z value` using estimate and its standard deviation:

```r
z <- glm_bal_inc$coefficients[3] / (summary(glm_bal_inc))$coefficients[3,2]
z
```

```
  income
4.174178
```

and then the p-value:

```
(1-pnorm(z))*2
```

```
      income
2.990638e-05
```

Using the hessian matrix obtained before as a side product of the Newton-Raphson algorithm, we retrieve comparable values with `glm` outputs for $\hat{\sigma}_j$:

```
std_errors <- sqrt(diag(solve(-as.matrix(sol$hessian))))
std_errors
```

```
 (Intercept)      balance       income
4.346349e-01 2.273110e-04 4.984579e-06
```

The `R` command to get confidence interval of estimators based on Wald statistic is the following (by default $\alpha = 5\%$)

```
confint.default(glm_bal_inc)
```

```
                    2.5 %        97.5 %
(Intercept) -1.239258e+01 -1.068836e+01
balance      5.201460e-03  6.092746e-03
income       1.103823e-05  3.057972e-05
```

We can retrieve it manually using coefficient estimate and standard deviation:

```
output_bal_inc = summary(glm_bal_inc)$coefficients
bal_std_estimate <- output_bal_inc[2,1]
bal_std_error <- output_bal_inc[2,2]

# upper bound for beta(balance) at 5%
upper <- bal_std_estimate + 1.96 * bal_std_error

# lower bound for beta(balance) at 5%
```

```r
lower <- bal_std_estimate - 1.96 * bal_std_error
(bal_confint <- c(lower, upper))
```

```
[1] 0.005201452 0.006092754
```

The following R command provides confidence interval of estimators using a more advance profile likelihood method:

```r
confint(glm_bal_inc)
```

```
                    2.5 %        97.5 %
(Intercept) -1.241910e+01 -1.071361e+01
balance      5.214030e-03  6.105971e-03
income       1.105359e-05  3.060844e-05
```

More details on profile likelihood method can be found here.

7.2.2 Tests on model coefficients

Based on the same idea, it is possible to test for the nullity of a subset of the model coefficients.

Denoting: H_0: $\beta_1 = \cdots = \beta_q = 0$, H_1: $\exists j \in \{1, \cdots, q\} \mid \beta_j \neq 0$, $\hat{\beta} = (\hat{\beta}_1, \cdots, \hat{\beta}_p)$ the MLE and $\hat{\beta}_{1:q} = (\hat{\beta}_1, \cdots, \hat{\beta}_q)$ the vector of first q parameters.

We have under H_0:

$$\hat{\beta}_{1:q}^T (X^T W_{\hat{\beta}} X)_{1:q}^{-1} \hat{\beta}_{1:q} \xrightarrow{\mathcal{L}} \chi_q^2$$

where $(X^T W_{\hat{\beta}} X)_{1:q}^{-1}$ is the $q \times q$ upper left block matrix extracted from the inverse of hessian.

We will reject H_0 at level α if the observed value $\hat{\beta}_{1:q}^T (X^T W_{\hat{\beta}} X)_{1:q}^{-1} \hat{\beta}_{1:q}$ is above the $1-\alpha$ quantile of the χ_q^2 distribution.

We show below the Wald tests for each coefficient in the model using `summary`:

```r
summary(glm_default)
```

```
Call:
glm(formula = default ~ ., family = "binomial", data = default_data)

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.087e+01  4.923e-01 -22.080  < 2e-16 ***
studentYes  -6.468e-01  2.363e-01  -2.738  0.00619 **
balance      5.737e-03  2.319e-04  24.738  < 2e-16 ***
income       3.033e-06  8.203e-06   0.370  0.71152
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1571.5  on 9996  degrees of freedom
AIC: 1579.5

Number of Fisher Scoring iterations: 8
```

These tests can be also performed in `R` using `car::Anova` or `aod::wald.test` routines. In particular when categorical variables have more than two levels these functions allow to test each variables as a whole (vs coefficient by coefficient when using `summary`)

```r
car::Anova(glm_default, type=3, test.statistic= "Wald")
```

```
Analysis of Deviance Table (Type III tests)

Response: default
            Df    Chisq Pr(>Chisq)
(Intercept)  1 487.5303  < 2.2e-16 ***
student      1   7.4947   0.006188 **
balance      1 611.9470  < 2.2e-16 ***
income       1   0.1368   0.711520
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can retrieve these outputs manually:

```r
sum_default <- summary(glm_default)
beta_income <- sum_default$coefficients[4,1]
```

```r
stdev_income <- sum_default$coefficients[4,2]

wald <- beta_income ^ 2 / stdev_income ^ 2

1-pchisq(wald, df = 1)
```

```
[1] 0.7115203
```

```r
z_val <- sum_default$coefficients[4,3]

2*(1-pnorm(abs(z_val)))
```

```
[1] 0.7115203
```

```r
# Testing the income coefficient (Terms = 4)
aod::wald.test(b = coef(glm_default), Sigma = vcov(glm_default), Terms = 4)
```

```
Wald test:
----------

Chi-squared test:
X2 = 0.14, df = 1, P(> X2) = 0.71
```

With all routines, the p-value for the income coefficient is 0.71 validating the null hypothesis.

Using `Terms` or `L` parameters in `aod::wald.test` it is also feasible to test the null hypothesis for a subsets of parameters:

```r
# Testing the income coefficient (Terms = 4)
aod::wald.test(b = coef(glm_default), Sigma = vcov(glm_default), Terms = 1:3)
```

```
Wald test:
----------

Chi-squared test:
X2 = 698.3, df = 3, P(> X2) = 0.0
```

The null hypothesis is rejected for the model with balance and student.

There are known issues with the Wald test:

- It is not invariant to re-reparametrisation. In the case of logistic regression we have a nonlinear model where the individual parameters are not of foremost interest, but rather odds ratios for instance, which are nonlinear transforms of the parameters.
- Hauck and Donner (1977) have shown that the Wald test has undesirable properties for logistic regression. In particular when $\hat{\beta}_j \to \infty$ (e.g. in case of separation or quasi separation), it is likely that $\hat{\sigma}_j \to \infty$. The result can be that the Wald statistic tends to zero as the distance between the parameter estimate and the null value increases (null hypothesis gets more and more wrong).

So in the context of Logistic Regression (and GLM), while Wald statistics are usually reported by statistical routines, the likelihood ratio or deviance-based tests are often favored.

7.3 Likelihood ratio tests

It is possible to test for the nullity of a subset of the model coefficients using Likelihood Ratio statistics.

Denoting: H_0: $\beta_1 = \cdots = \beta_q = 0$, H_1: $\exists j \in \{1, \cdots, q\} \mid \beta_j \neq 0$, and $\hat{\beta} = (\hat{\beta}_1, \cdots, \hat{\beta}_p)$ the MLE, we have under H_0:

$$-2\left(\ell_{\mathrm{H}_0}(Y, \hat{\beta}_{\mathrm{H}_0}) - \ell(Y, \hat{\beta})\right) \xrightarrow{\mathcal{L}} \chi^2_q$$

where $\ell_{\mathrm{H}_0}(Y, \hat{\beta}_{\mathrm{H}_0})$ is the log-likelihood of:

$$\mathrm{logit}(p_\beta(X)) = x_{q+1}\beta_{q+1} + \cdots + x_n\beta_n$$

Consider two models, a larger model with l parameters and likelihood L_L and a smaller model with s parameters and likelihood L_S, where the smaller model represents a subset of the larger model. Typically the smaller model is equivalent to the large model where we have imposed:

$$\mathrm{H}_0: \quad \beta_j = ... = \beta_{j+r} = 0$$

Likelihood Ratio tests on variables may be performed in `R` using `car::Anova`:

```
car::Anova(glm_default, type=3, test.statistic= "LR")
```

```
Analysis of Deviance Table (Type III tests)

Response: default
        LR Chisq Df Pr(>Chisq)
student     7.42  1   0.006445 **
balance  1335.95  1  < 2.2e-16 ***
income      0.14  1   0.711514
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Using base `R` `anova` it is also possible to test subsets of variables and in particular individual variables within the "full" model:

```r
glm_wo_student <- glm(default ~ balance + income,
                      data = default_data,
                      family = "binomial")
glm_wo_balance <- glm(default ~ student + income,
                      data = default_data,
                      family = "binomial")
glm_wo_income <- glm(default ~ student + balance,
                      data = default_data,
                      family = "binomial")
```

```r
anova(glm_wo_student, glm_default, test = "LRT")
```

```
Analysis of Deviance Table

Model 1: default ~ balance + income
Model 2: default ~ student + balance + income
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      9997     1579.0
2      9996     1571.5  1   7.4214 0.006445 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

We can retrieve this result manually:

```r
LRT <- 2 * (logLik(glm_default)-logLik(glm_wo_student))

1 - pchisq(LRT, df = 1)
```

```
'log Lik.' 0.006445112 (df=4)
```

```r
anova(glm_wo_balance, glm_default, test = "LRT")
```

```
Analysis of Deviance Table

Model 1: default ~ student + income
Model 2: default ~ student + balance + income
  Resid. Df Resid. Dev Df Deviance  Pr(>Chi)
1      9997     2907.5
2      9996     1571.5  1     1336 < 2.2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
anova(glm_wo_income, glm_default, test = "LRT")
```

```
Analysis of Deviance Table

Model 1: default ~ student + balance
Model 2: default ~ student + balance + income
  Resid. Df Resid. Dev Df Deviance Pr(>Chi)
1      9997     1571.7
2      9996     1571.5  1  0.13677   0.7115
```

The deviance defined as $D = -2\ell$ is often reported by statistical software in place of log-likelihood. A large likelihood corresponding to a small deviance.

A better coverage of tests in the context of Logistic Regression can be found here or in (Hosmer et al., 2013). See also here for a data analysis using `R` and see the question here for a very detailed description of the outputs of `glm()` (in particular this answer).

7.4 Goodness of Fit test / Calibration

Although it is generally not recommended by practitioners and theoreticians, the Hosmer & Lemeshow test (see here, here or here) allows to quickly assess the "goodness of fit" of a Logistic Regression.

But more than the test in itself, the underlying motivation is interesting: the Logistic Regression model provides an estimate of the probability of an outcome (success/failure, here the

default is success or 1). The estimated probability of this outcome should be close to the true observed probability.

The Hosmer & Lemeshow test assess if observed event rates match expected event rates in subgroups of "similar" observations. Models for which expected and observed event rates agree on these subgroups are considered well calibrated.

A first step of the test is to order the predicted probabilities of the outcome and divide it into Q groups (usually using deciles, Q=10).

Then the average predicted probability for each group is computed and compared to the observed probability.

The Hosmer & Lemeshow test statistic H is compared to a χ^2_{Q-2} distribution:

$$H = \sum_{q=1}^{Q} \frac{(o_q - m_q\mu_q)^2}{m_q\mu_q(1-\mu_q)}$$

where:

- o_q denotes the number of success ($Y = 1$) observed in group q,
- μ_q denotes the mean of $p_{\hat{\beta}}(x_i)$ in group q,
- m_q denotes the number of observations in group q, so that $m_q\mu_q$ is the expected number of success in group q.

The null hypothesis is that observed/expected outcomes are close along all subgroups.

```r
library(glmtoolbox)
hltest(glm_default)
```

```
  The Hosmer-Lemeshow goodness-of-fit test

 Group Size Observed     Expected
     1 1000        0   0.02653992
     2 1000        0   0.10737240
     3 1000        0   0.29143249
     4 1000        1   0.67265778
     5 1000        2   1.39515666
     6 1000        1   2.87108745
     7 1000        7   5.98948667
     8 1000       16  13.74542953
     9 1000       45  39.52811751
    10 1000      261 268.37271994
```

```
        Statistic =  3.68229
degrees of freedom =  8
           p-value =  0.88459
```

Here the p-value for a chi-squared statistic of $H = 3.68$ with $df = Q - 2 = 8$ is $p = 0.885$ which is well above the usual levels (eg 0.05), so that the null hypothesis is accepted, goodness of fit is acceptable.

However the Hosmer & Lemeshow test is dependent on the choice of Q and the binning performed on probabilities and is sometimes considered unreliable.

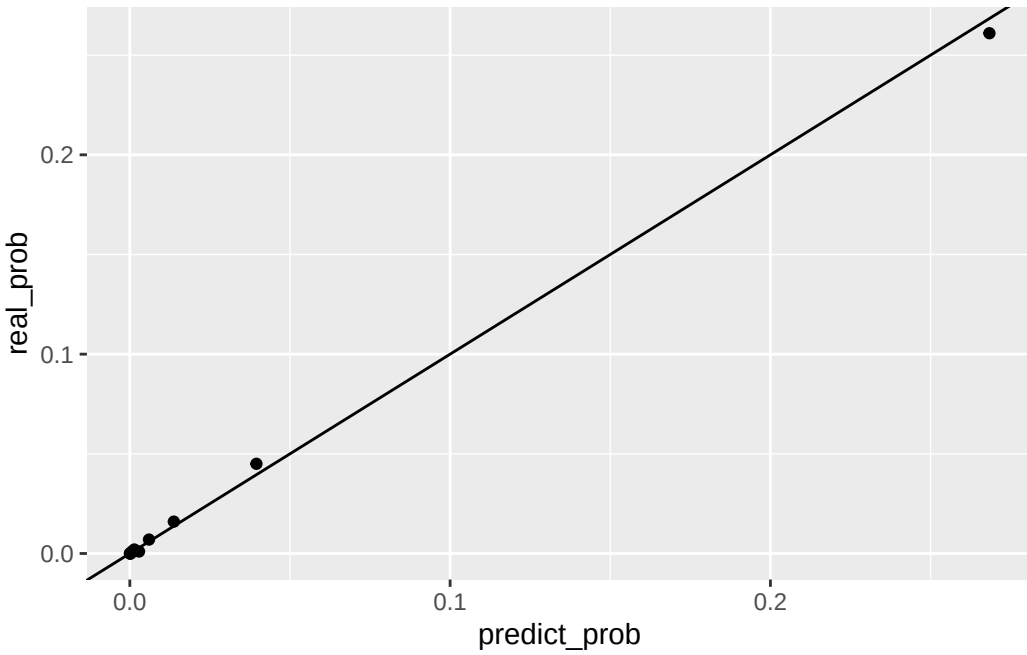
Nonetheless it is usual to assess or diagnose the good calibration of a model probabilities using Calibration Plots or Probability Calibration Curves (see here for a recent R package from the tidyverse/tidymodel ecosystem and here for a scikit-learn version): they are used to visualize if predictions are consistent with the observed event rates (be it on the training set or a testing set, which is better). For example considering the `default` data set we have:

```r
check_default_prob <- as_tibble(
  cbind(fitted=glm_default$fitted.values,
        Y = default_data %>%
              mutate(default = if_else(default == "Yes", 1, 0)) %>%
                                  pull(default)))
(calibration_data <- check_default_prob %>%
  mutate(bins_prob = cut(fitted,
                         breaks = quantile(fitted,seq(0,1,0.10)),
                         include.lowest = TRUE)) %>%
  group_by(bins_prob) %>%
  summarize(n = n(),
            def = sum(Y),
            no_def = n - def,
            predict_prob = mean(fitted),
            real_prob = def/n,
            forecast_acc = def / sum(check_default_prob$Y)))
```

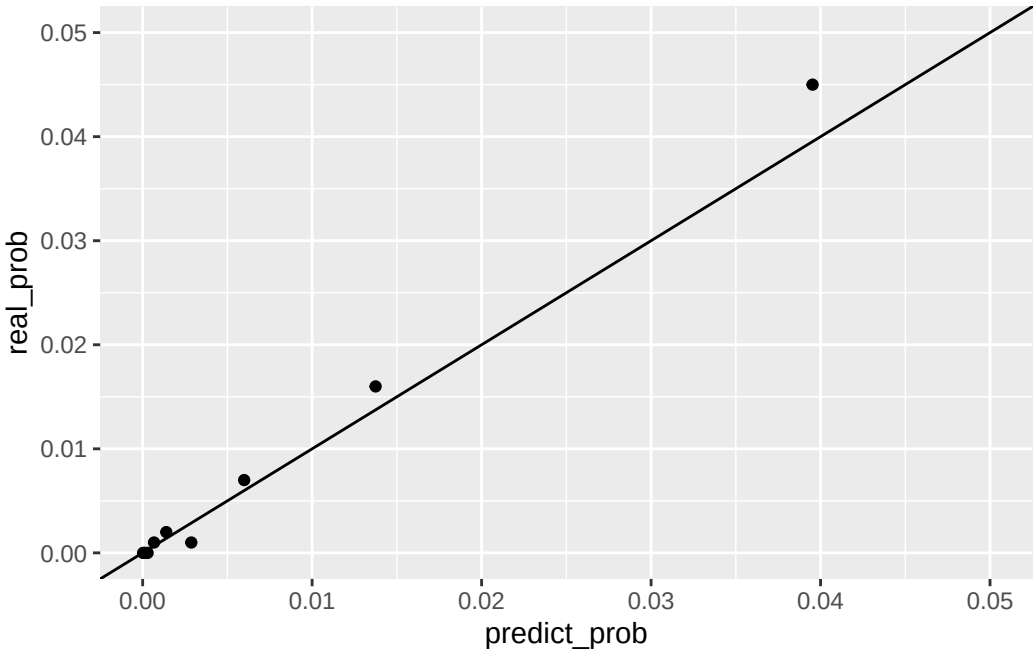
```
# A tibble: 10 x 7
   bins_prob               n   def no_def predict_prob real_prob forecast_acc
   <fct>               <int> <dbl>  <dbl>        <dbl>     <dbl>        <dbl>
 1 [1.03e-05,5.14e-05]  1000     0   1000    0.0000265     0          0
 2 (5.14e-05,0.000176]  1000     0   1000    0.000107      0          0
 3 (0.000176,0.000443]  1000     0   1000    0.000291      0          0
 4 (0.000443,0.000945]  1000     1    999    0.000673      0.001      0.00300
 5 (0.000945,0.00197]   1000     2    998    0.00140       0.002      0.00601
 6 (0.00197,0.00402]    1000     1    999    0.00287       0.001      0.00300
```

```
 7 (0.00402,0.0088]      1000      7     993     0.00599        0.007       0.0210
 8 (0.0088,0.021]        1000     16     984     0.0137         0.016       0.0480
 9 (0.021,0.0709]        1000     45     955     0.0395         0.045       0.135
10 (0.0709,0.978]        1000    261     739     0.268          0.261       0.784
```

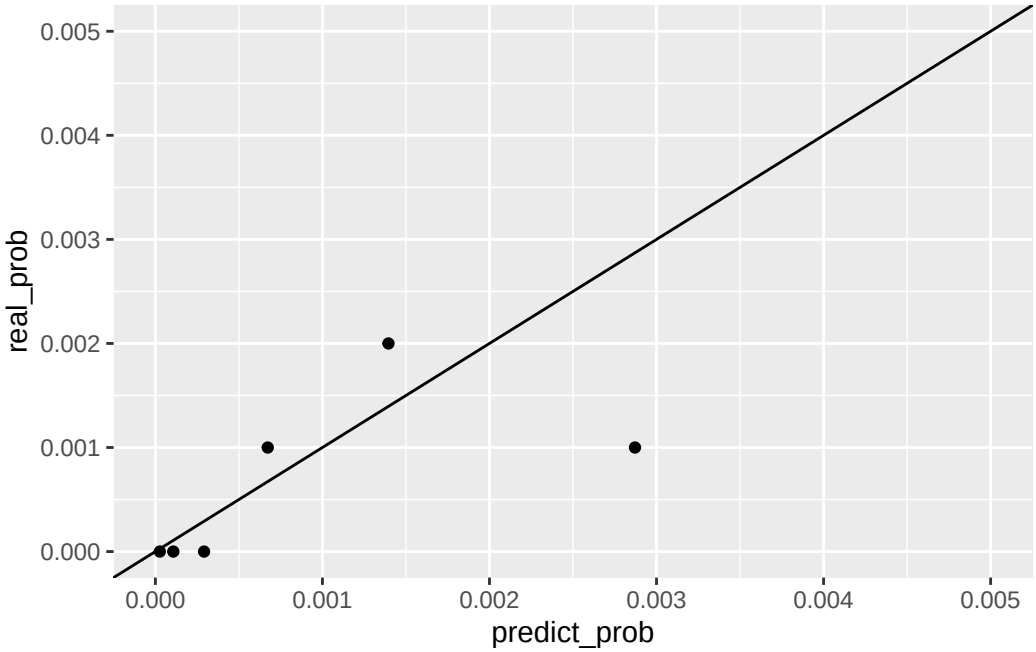
```r
(calib_plot <- ggplot(calibration_data, aes(x = predict_prob, y = real_prob)) +
  geom_point() +
  geom_abline())
```




```r
calib_plot + coord_cartesian(xlim=c(0, 0.05), ylim=c(0, 0.05))
```




```
calib_plot + coord_cartesian(xlim=c(0, 0.005), ylim=c(0, 0.005))
```



The model tends to slightly overestimate/underestimates some deciles without a clear pattern.

8 Variable selection, model assessment

We have seen in the last section and in the context of statistical inference how to compare two or more nested Logistic regression models (typically a reduced model with less variables than a reference or full model).

In this section we places ourselves in the context of statistical learning, with an emphasis on prediction.

Real life data sets usually contain a large number of predictors or inputs. As their number grow, the analyst will have to analyse a growing number of possible models or variable combinations (typically 2^p where p denotes the number of predictors). Furthermore as the number of predictors grow the models might overfit the training set, causing a deterioration of prediction error.

Variable selection is a critical aspect of building Logistic Regression models and prediction models in general. This process aims to find models that are both parsimonious and predictive. Parsimony, in this context, implies the selection of a minimal set of predictor variables that still provides an accurate representation of the data, avoiding overfitting while enhancing model generalization.

This is a complex subject that we will not cover in depth in this course.

For a better coverage, chapter 7 `Model Assessment and Selection` of Hastie et al. (2009) discusses in depth the interplay between bias, variance and model complexity, in a general setting. Chapter 5 `Resampling Methods` of James et al. (2021) introduces in particular cross-validation as a tool to estimate the test error associated with a given statistical learning method in order to assess its performance, or to select the methods appropriate level of flexibility (or hyperparameters). Chapter 6 `Linear Model Selection and Regularization` discusses methods to "automatically" perform variable selection in the context of linear models.

In this section we consider some common methods for selecting subsets of predictors that you might encounter or be asked to perform: this include best subset, stepwise model selection procedures and penalization or shrinkage. We use the Default and Mixture data sets as examples.

8.1 Best subset

A first common approach is the "best subset" method, which evaluates all possible combinations of predictor variables, resulting in $2^p - 1$ models for p predictors, and selects the one that optimizes a specified criterion, such as the Akaike Information Criterion (AIC) or

Bayesian Information Criterion (BIC). It is computationally intensive and usually restricted to low dimension data sets.

Algorithm: Best subset selection

---

1. For $k = 1, \cdots, p$:
   (a) Fit all $\binom{p}{k}$ models that contain exactly k predictors.
   (b) Pick the best among these $\binom{p}{k}$ models, and call it $\mathcal{M}_k$. In the case of Logistic Regression, best usually means largest log-likelihood or min deviance.
2. Select a single best model from among $\mathcal{M}_1, \cdots, \mathcal{M}_p$ using a criterion. Usually the prediction error on a validation set, AIC, BIC…

Given $|\mathcal{M}|$ we define the two following criteria which represent two way of penalizing the maximised log-likelihood $\ell(Y, \hat{\beta})$:

$$\text{AIC}(\mathcal{M}) = -2\ell(Y, \hat{\beta}) + 2|\mathcal{M}|$$

and

$$\text{BIC}(\mathcal{M}) = -2\ell(Y, \hat{\beta}) + |\mathcal{M}| \log(n)$$

The package `bestglm` allows best subset selection up to roughly 15 variables, by default it uses BIC. We use the Default data set to illustrate because Agriculture Farm Lending has around 30 variables:

```
library(bestglm)

default_best <- as.data.frame(default_data %>%
                              mutate(Y=if_else(default=='Yes', 1, 0)) %>%
                              select(-default))
# p must be < 15 for GLM
# Error in bestglm(as.data.frame(don_desbois_quanti), family = binomial) :
# p = 22. must be <= 15 for GLM.
```

```r
mod_sel <- bestglm(default_best, family = binomial)
mod_sel
```

```
BIC
BICq equivalent for q in (0.000417890560228229, 0.989405669357382)
Best Model:
                 Estimate  Std. Error    z value      Pr(>|z|)
(Intercept) -10.749495878 0.369191361 -29.116326 2.230782e-186
studentYes   -0.714877620 0.147519010  -4.846003  1.259734e-06
balance       0.005738104 0.000231847  24.749526 3.136911e-135
```

8.2 Stepwise Logistic Regression

Alternatively, stepwise methods, including forward and backward selection, iteratively add or remove variables based on the chosen criterion. These approaches, while useful, can be also computationally intensive, especially for high-dimensional datasets.

Algorithm: Forward stepwise selection

---

1. For $k = 1, \cdots, p$:
   (a) Consider all $p+1-k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor.
   (b) Pick the best among these $p+1-k$ models, and call it $\mathcal{M}_{k+1}$. In the case of Logistic Regression, best usually means largest log-likelihood or min deviance.
2. Select a single best model from among $\mathcal{M}_1, \cdots, \mathcal{M}_p$ using a criterion. Usually the prediction error on a validation set, AIC, BIC…

---

or

Algorithm: Backward stepwise selection

---

1. Let $\mathcal{M}_p$ denote the full model, which contains all p predictors.
2. For $k = p, \cdots, 1$:
   (a) Consider all k models that contain all but one of the predictors in $\mathcal{M}_k$, for a total of $k-1$ predictors.
   (b) Pick the best among these k models, and call it $\mathcal{M}_{k-1}$. In the case of Logistic Regression, best usually means largest log-likelihood or min deviance.
3. Select a single best model from among $\mathcal{M}_1, \cdots, \mathcal{M}_p$ using a criterion. Usually the prediction error on a validation set, AIC, BIC…

Additionally versions mixing forward and backward stepwise selection exist. For example adding variables to the model sequentially, in the same way as forward selection then at each step the method tries remove variables that no longer provide a significant improvement in the model fit.

We give below the `R` command to perform forward/backward

```
# define intercept-only model
intercept_only <- glm(Y ~ 1,
                      data=default_data %>%
                      mutate(Y=if_else(default=='Yes', 1, 0)) %>%
                      select(-default), family="binomial")

# define model with all predictors
all <- glm(Y ~ .,
           data=default_data %>%
           mutate(Y=if_else(default=='Yes', 1, 0)) %>%
           select(-default), family="binomial")
```

```
# perform forward stepwise regression based on AIC (k=2)
forward_aic <- step(intercept_only,
                    direction='forward',
                    scope=formula(all),
                    k=2,
                    trace = TRUE)
```

```
Start:  AIC=2922.65
Y ~ 1
```

```
          Df Deviance    AIC
+ balance  1   1596.5 1600.5
+ student  1   2908.7 2912.7
+ income   1   2916.7 2920.7
<none>         2920.7 2922.7

Step:  AIC=1600.45
Y ~ balance

          Df Deviance    AIC
+ student  1   1571.7 1577.7
+ income   1   1579.0 1585.0
<none>         1596.5 1600.5

Step:  AIC=1577.68
Y ~ balance + student

         Df Deviance    AIC
<none>        1571.7 1577.7
+ income  1   1571.5 1579.5
```

```r
summary(forward_aic)
```

```
Call:
glm(formula = Y ~ balance + student, family = "binomial", data = default_data %>%
    mutate(Y = if_else(default == "Yes", 1, 0)) %>% select(-default))

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.075e+01  3.692e-01 -29.116  < 2e-16 ***
balance      5.738e-03  2.318e-04  24.750  < 2e-16 ***
studentYes  -7.149e-01  1.475e-01  -4.846 1.26e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1571.7  on 9997  degrees of freedom
AIC: 1577.7
```

```
Number of Fisher Scoring iterations: 8
```

```r
# perform backward stepwise regression based on BIC (k=log(n))
backward_bic <- step(all,
                     direction='backward',
                     scope=formula(all),
                     k=log(nrow(default_data)),
                     trace = TRUE)
```

```
Start:  AIC=1608.39
Y ~ student + balance + income

          Df Deviance    AIC
- income   1   1571.7 1599.3
- student  1   1579.0 1606.6
<none>         1571.5 1608.4
- balance  1   2907.5 2935.1

Step:  AIC=1599.31
Y ~ student + balance

          Df Deviance    AIC
<none>         1571.7 1599.3
- student  1   1596.5 1614.9
- balance  1   2908.7 2927.1
```

```r
summary(backward_bic)
```

```
Call:
glm(formula = Y ~ student + balance, family = "binomial", data = default_data %>%
    mutate(Y = if_else(default == "Yes", 1, 0)) %>% select(-default))

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.075e+01  3.692e-01 -29.116  < 2e-16 ***
studentYes  -7.149e-01  1.475e-01  -4.846 1.26e-06 ***
balance      5.738e-03  2.318e-04  24.750  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

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

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1571.7  on 9997  degrees of freedom
AIC: 1577.7

Number of Fisher Scoring iterations: 8
```

Using the `keep` argument allows to keep each consecutive fitted `glm` models until the stopping criterion is met (see here):

```r
forward_aic <- step(intercept_only,
                    direction='forward',
                    scope=formula(all),
                    keep = dplyr::lst,
                    # keep = function(fit, aic) list(fit = fit, aic = aic),
                    k=2,
                    trace = TRUE)
```

```
Start:  AIC=2922.65
Y ~ 1

          Df Deviance    AIC
+ balance  1   1596.5 1600.5
+ student  1   2908.7 2912.7
+ income   1   2916.7 2920.7
<none>         2920.7 2922.7

Step:  AIC=1600.45
Y ~ balance

          Df Deviance    AIC
+ student  1   1571.7 1577.7
+ income   1   1579.0 1585.0
<none>         1596.5 1600.5

Step:  AIC=1577.68
Y ~ balance + student

         Df Deviance    AIC
<none>        1571.7 1577.7
```

```
+ income  1   1571.5 1579.5
```

It then allows to extract each sequential models, which can be useful for example to assess each

```r
forward_aic_2 <- forward_aic$keep[["fit", 2]]
summary(forward_aic_2)
```

```
Call:
glm(formula = Y ~ balance, family = "binomial", data = default_data %>%
    mutate(Y = if_else(default == "Yes", 1, 0)) %>% select(-default))

Coefficients:
              Estimate Std. Error z value Pr(>|z|)
(Intercept) -1.065e+01  3.612e-01  -29.49   <2e-16 ***
balance      5.499e-03  2.204e-04   24.95   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 2920.6  on 9999  degrees of freedom
Residual deviance: 1596.5  on 9998  degrees of freedom
AIC: 1600.5

Number of Fisher Scoring iterations: 8
```

8.3 Introducing penalized logistic regressions

As an alternative, penalized regression techniques like the Lasso (Least Absolute Shrinkage and Selection Operator) or Ridge offer an efficient means of variance reduction and/or variable selection by introducing a penalty term in the Logistic Regression estimation algorithm. In particular Lasso promotes the sparsity of coefficients and automatically selects relevant predictors while shrinking others to zero.

We remind that to estimate Logistic Regression parameters, we maximized in β the log-likelihood:

$$\ell(Y, \beta) = \log L(Y, \beta) = \sum_{i=1}^{n} \left(y_i \log(p_\beta(x_i)) + (1 - y_i) \log(1 - p_\beta(x_i)) \right)$$

In the context of penalized Logistic Regression the idea is to minimize in β:

$$-\ell(Y,\beta) + \lambda_2\|\beta\|_2 + \lambda_1\|\beta\|_1$$

where λ_1, λ_2 are two constants.

The underlying idea is to constraint or shrink the size of the coefficients estimates.

When $\lambda_1 > 0, \lambda_2 = 0$ we are using a Lasso penalty.

When $\lambda_1 = 0, \lambda_2 > 0$ we are using a Ridge penalty.

When $\lambda_1 > 0, \lambda_2 > 0$ we are using an Elastic Net penalty.

Without entering to much details in this section, Ridge does a proportional shrinkage of all coefficients while Lasso tends to truncates some of the coefficients at zero. Usually the following graphs are given in textbooks/slides to give some intuition about Ridge/Lasso (the figure below is taken from the original Lasso article by Tibshirani published in 1996):

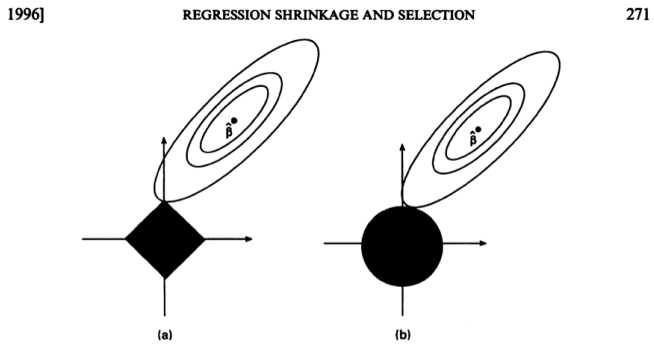


Fig. 2. Estimation picture for (a) the lasso and (b) ridge regression

We use here the Mixture data set to give further intuition about Ridge/Lasso penalties:

```
data_mixture_example <- readRDS("./presentation_data/data_mixture_example.rds")
y = as.numeric(data_mixture_example$Y)-1
X = as.matrix(data_mixture_example %>% select(x1,x2))
```

Ridge estimator is:

$$\hat{\beta}_{ridge} = \underset{\beta}{\text{argmin}} - \ell(Y,\beta) + \lambda\|\beta\|_2$$

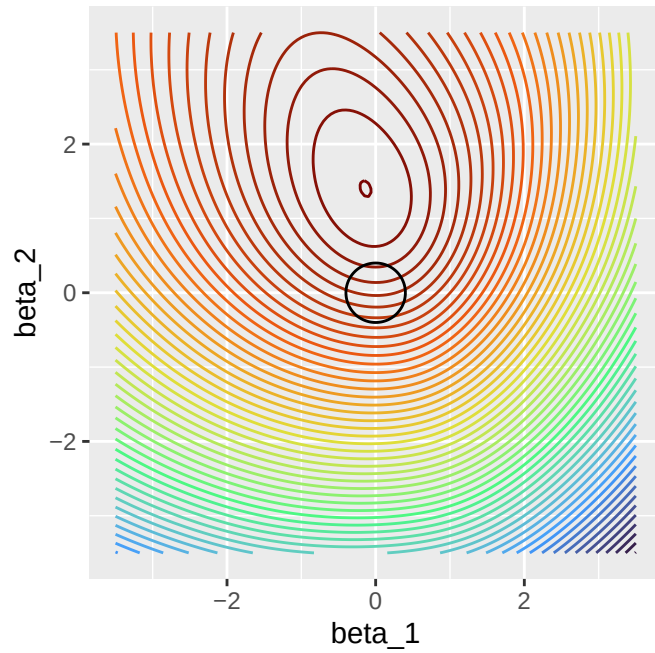
which rewrites:

$$\hat{\beta}_{ridge} = \underset{\beta}{\text{argmin}} - \ell(Y, \beta)$$

subject to $\|\beta\|_2 \leq t$, where t maps to λ.

We plot below the log-likelihood of the Logistic Regression (colored levels lines centered around the MLE) for the Mixture data set, adding the Ridge constraint (the circle in black is the boundary $\|\beta\|_2 \leq t$) :

```
grid <- expand.grid(beta_1 = seq(-3.5, 3.5, .05),
                    beta_2 = seq(-3.5, 3.5, .05)) %>%
         as_tibble()
# Prediction function used to classify areas on the grid and imply the decision boundary
LL  <- function(b0, b1, b2){
  beta <-  c(b1, b2)
  return( sum(-y*log(1 + exp(-(b0+X%*%beta))) -
  (1-y)*log(1 + exp(b0+X%*%beta))))
}

LL_V <- Vectorize(LL)

# extract intercept coeff from glm (beta0 is not varying here)
mixture_glm <- glm(Y ~ ., data=data_mixture_example, family="binomial")
beta0 <- mixture_glm$coefficients[1]

grid <- grid %>% mutate(beta_0 = beta0, LL = LL_V(beta_0, beta_1, beta_2))

xc <- 0
yc <- 0
r <- 0.4

ridge_plot <- ggplot(grid) +
    geom_contour(aes(x = beta_1, y = beta_2, z = LL,
                     colour = after_stat(level)), binwidth = 15) +
    scale_color_viridis_c(option="H") +
    annotate("path",
             x=xc+r*cos(seq(0,2*pi,length.out=100)),
             y=yc+r*sin(seq(0,2*pi,length.out=100))) +
     theme(legend.position="none")

(ridge_plot + coord_fixed())
```

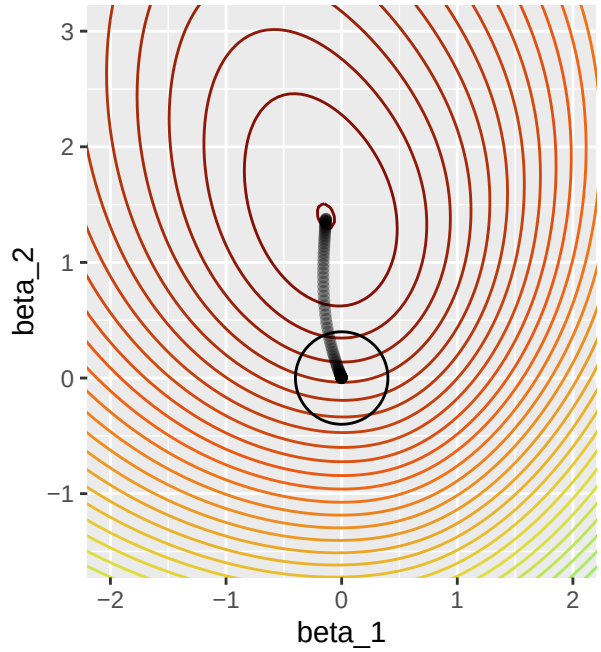


We obtain Ridge Logistic Regression parameters for x_1, x_2 as a function of λ:

```
library(glmnet)
library(glmnetUtils) # convenient package allowing to use R formulas instead of glmnet spa

mixture_ridge <- glmnetUtils::glmnet(Y ~ .,
                                     data=data_mixture_example,
                                     family="binomial",
                                     alpha=0, # ridge penalty
                                     lambda.min.ratio=0.000001)

ridge_result <- as_tibble(as.matrix(
                    cbind(mixture_ridge$lambda,
                          mixture_ridge$a0,
                          t(mixture_ridge$beta))))
names(ridge_result) <-  c("lambda", "(Intercept)", row.names(mixture_ridge$beta))
```

We then visualize the Ridge Logistic Regression parameters β_1, β_2 path as λ increases. It starts from the Logistic Regression MLE for $\lambda = 0$ and it is then shrunken "uniformly" towards zero as λ increases.

```
ridge_plot +
  geom_point(data=ridge_result, aes(x=x1, y=x2), alpha=0.3) +
  coord_fixed(xlim = c(-2, 2), ylim = c(-1.5, 3))
```

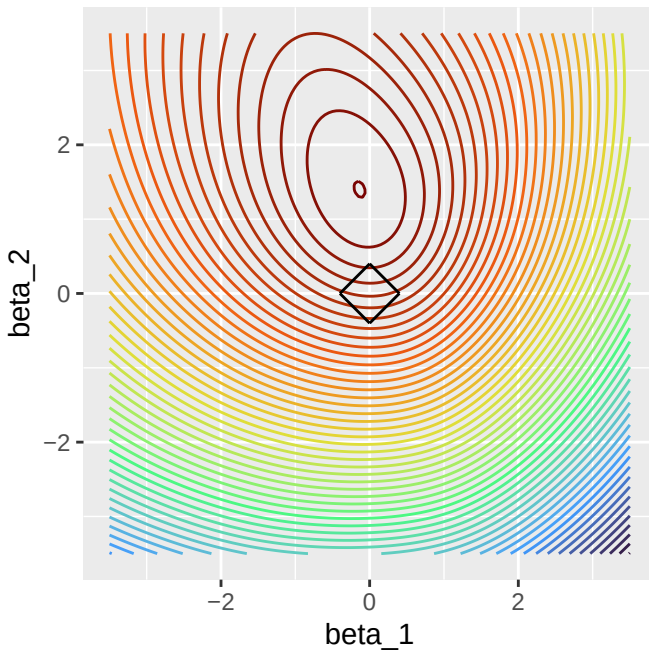


Exercise : implement Newton Raphson or IRLS for Ridge constraint (using either gradient/hessian of penalized likelihood or closed form solution for Linear Model with Ridge)

We plot below the log-likelihood of the Logistic Regression (colored levels lines centered around the MLE) for the Mixture data set, adding the Lasso constraint (the diamond/square in black is the boundary $\|\beta\|_1 \leq t$) :

```
grid <- expand.grid(beta_1 = seq(-3.5, 3.5, .05),
                    beta_2 = seq(-3.5, 3.5, .05)) %>%
          as_tibble()
# Prediction function used to classify areas on the grid and imply the decision boundary

grid <- grid %>% mutate(beta_0 = beta0, LL = LL_V(beta_0, beta_1, beta_2))

h <- 0.4

lasso_plot <- ggplot(grid) +
```

```
    geom_contour(aes(x = beta_1, y = beta_2, z = LL,
                     colour = after_stat(level)), binwidth = 15) +
    scale_color_viridis_c(option="H") +
    annotate("segment", x=-h, xend=0, y=0, yend=h, col = 'black') +
    annotate("segment", x=0, xend=h, y=h, yend=0, col = 'black') +
    annotate("segment", x=h, xend=0, y=0, yend=-h, col = 'black') +
    annotate("segment", x=0, xend=-h, y=-h, yend=0, col = 'black') +
    theme(legend.position="none")

(lasso_plot + coord_fixed())
```



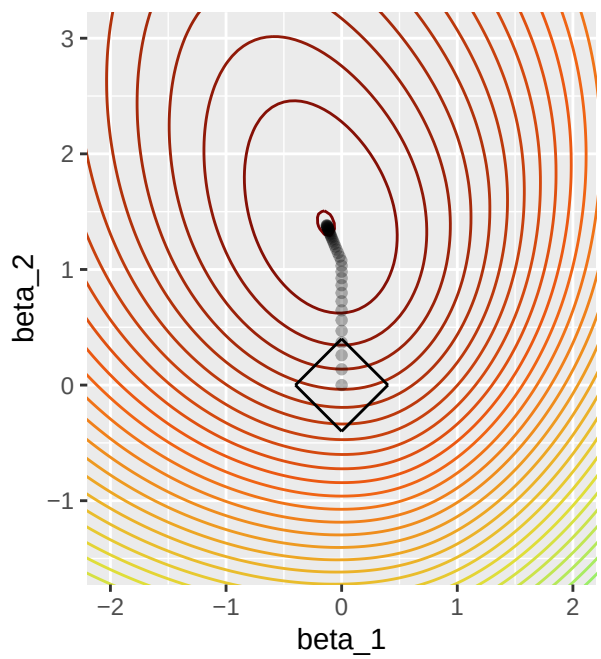
We obtain Lasso Logistic Regression parameters for x_1, x_2 as a function of λ:

```
mixture_lasso <- glmnetUtils::glmnet(Y ~ .,
                                     data=data_mixture_example,
                                     family="binomial",
                                     alpha=1,
                                     lambda.min.ratio=0.000001)

lasso_result <- as_tibble(as.matrix(
                              cbind(mixture_lasso$lambda,
```

```
                                        mixture_lasso$a0,
                                        t(mixture_lasso$beta))))
names(lasso_result) <-  c("lambda", "(Intercept)", row.names(mixture_lasso$beta))
```

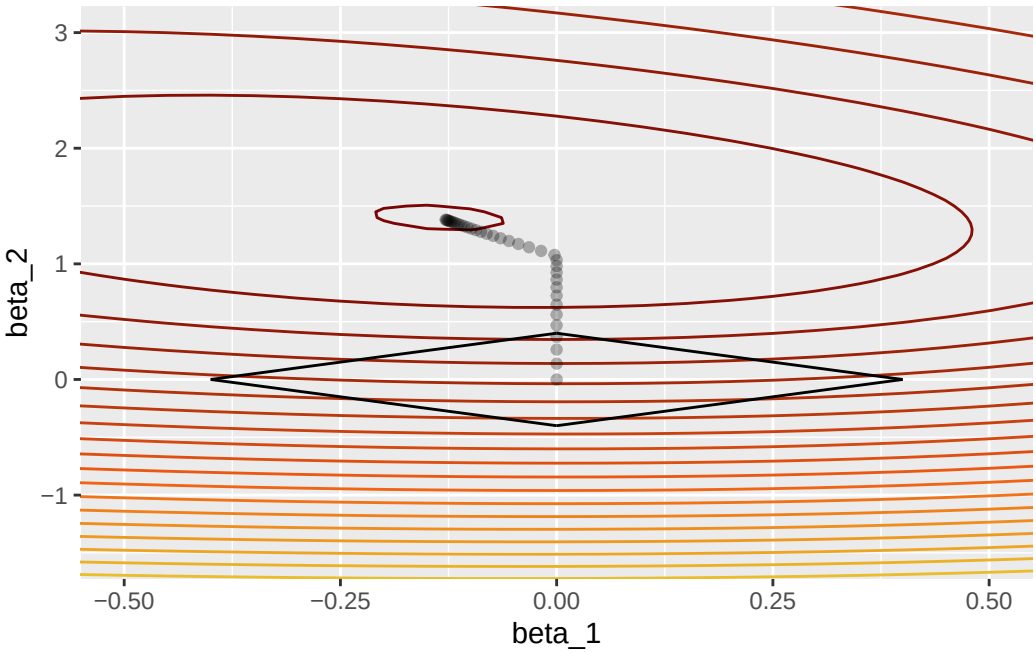
We then visualize the Lasso Logistic Regression parameters β_1, β_2 path as λ increases. It starts from the Logistic Regression MLE for $\lambda = 0$ and as λ increases and at some point the solution of the constrained optimization is likely to occur at one of the corners of the diamond, which is indeed the case with the Mixture data set, the β_1 parameter being first shrunk to zero:

```
lasso_plot +
  geom_point(data=lasso_result, aes(x=x1, y=x2), alpha=0.3) +
  coord_fixed(xlim = c(-2, 2), ylim = c(-1.5, 3))
```



Stretching the graph a little bit to better show the Lasso path:

```
lasso_plot +
  geom_point(data=lasso_result, aes(x=x1, y=x2), alpha=0.3) +
  coord_cartesian(xlim = c(-0.5, 0.5), ylim = c(-1.5, 3))
```



In R the package `glmnet` implements the Lasso, Ridge and Elastic Net penalties in particular for Logistic Regression.

The following paragraph in the documentation helps to compare `glmnet` with other packages that might be useful for reproducibility purposes.

8.4 Model assessment

We have seen in the first lesson that in the context of statistical learning the approach was to estimate a classifier or Score with the best possible Risk or any other metric such as the AUC of a ROC curve. The final goal being to predict unobserved outputs given unobserved inputs having trained a classifier using the data at hands.

To assess how well the classifier or Score will "generalize" to new data, a practical approach is to split the data set into a training set and a test set or validation set. The training set being used to learn the classifier or Score, the validation set to estimate the generalization error of the classifier.

8.4.1 Introductory example

We re-use here the Mixture example from Hastie et al. (2009), comparing Logistic regressions fitted using polynomial functions of the predictors (as performed in James et al. (2021), chapter

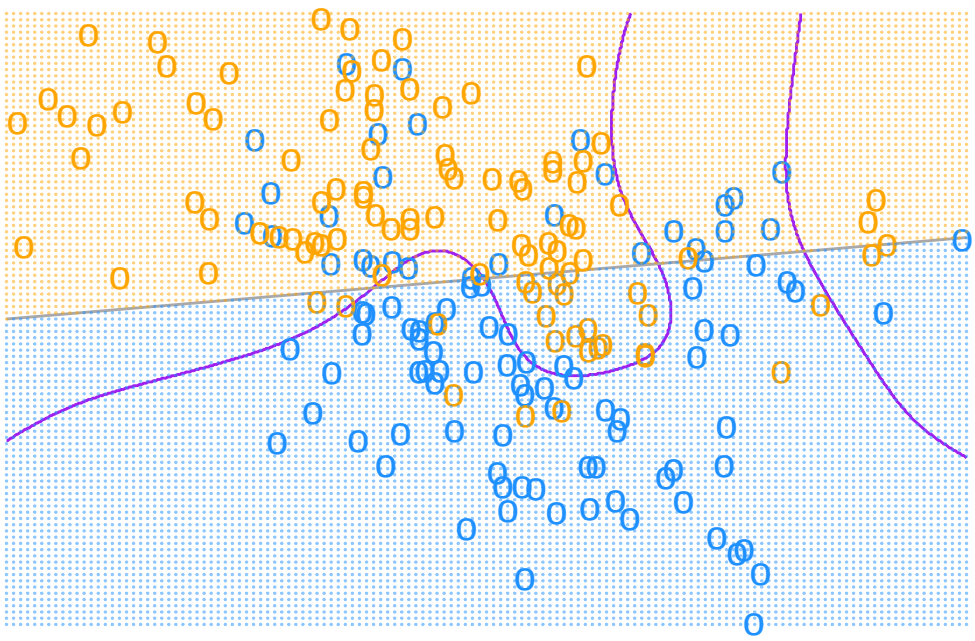
5.1.5 Cross-Validation on Classification Problems, on a modified version of the Mixture data set, with much lower Bayes error, 13% vs 20% in original data set).

Since we know the generating distribution, we have been able to simulate a very large testing set (10,000 observations per class), allowing to estimate the Bayes error, and the "true" testing error for any proposed classifier (see first lesson, for decision boundaries of various classifiers).

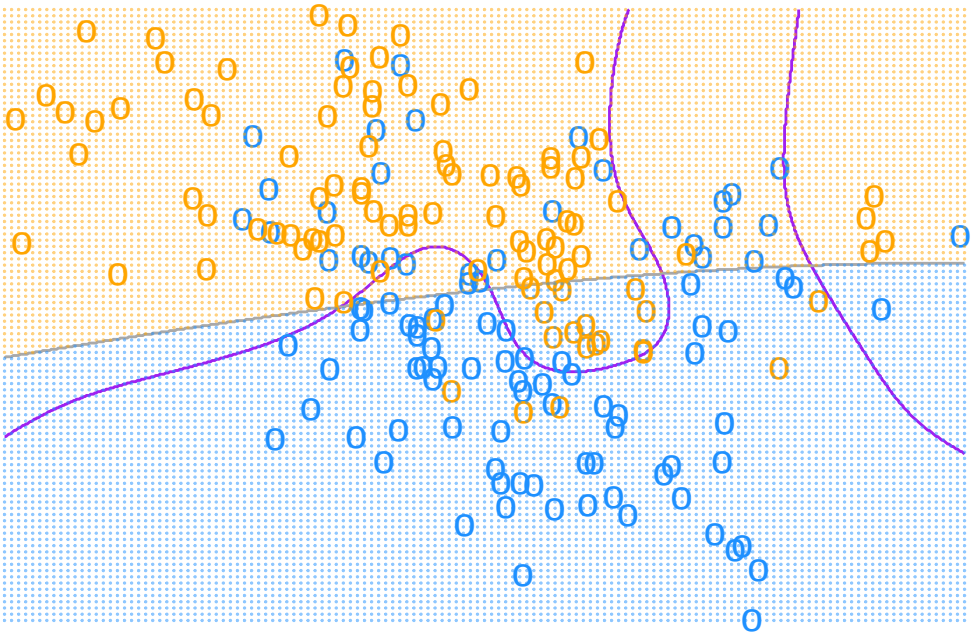
However in a real life setting, we usually have limited data (for example in our case we assume we only have the original data set with only 100 observations per class. In the absence of a very large test set that can be used to directly estimate the "true" testing error, a number of techniques are used to estimate this quantity using the available training data. We a class of methods called resampling methods, that estimate the test error rate by holding out a subset of the training observations from the fitting process, and then applying the statistical learning method to those held out observations.

Here, for illustrating purposes, the learning method is a polynomial Logistic regression, we vary below the degree d of polynomial:

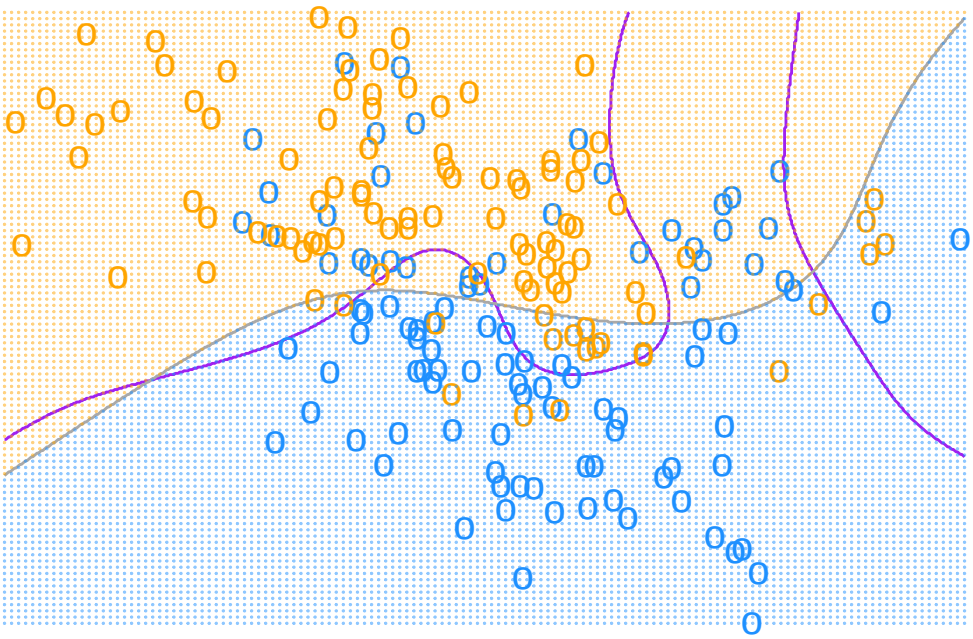
- Degree 1:



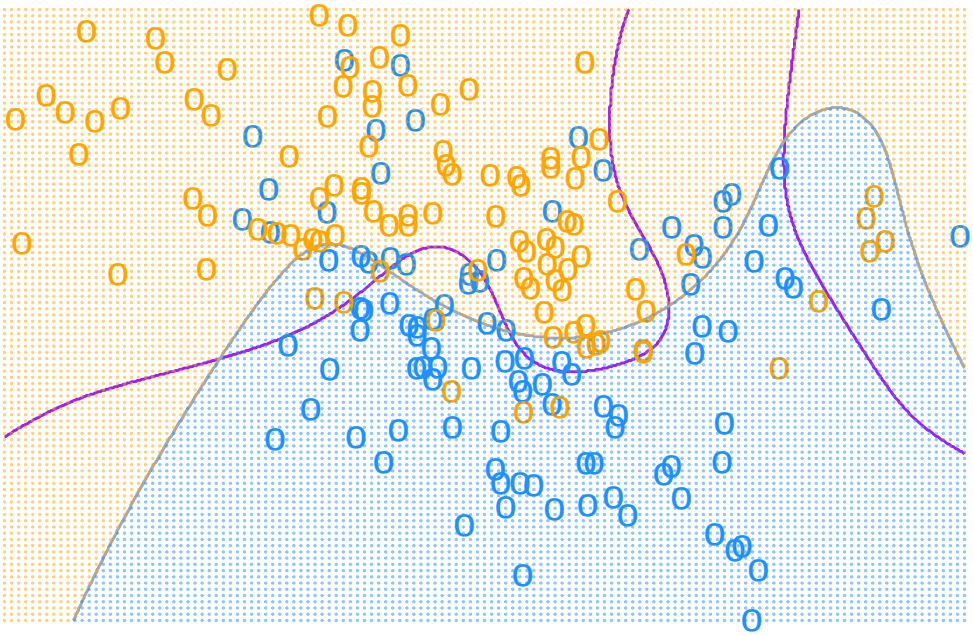
- Degree 2



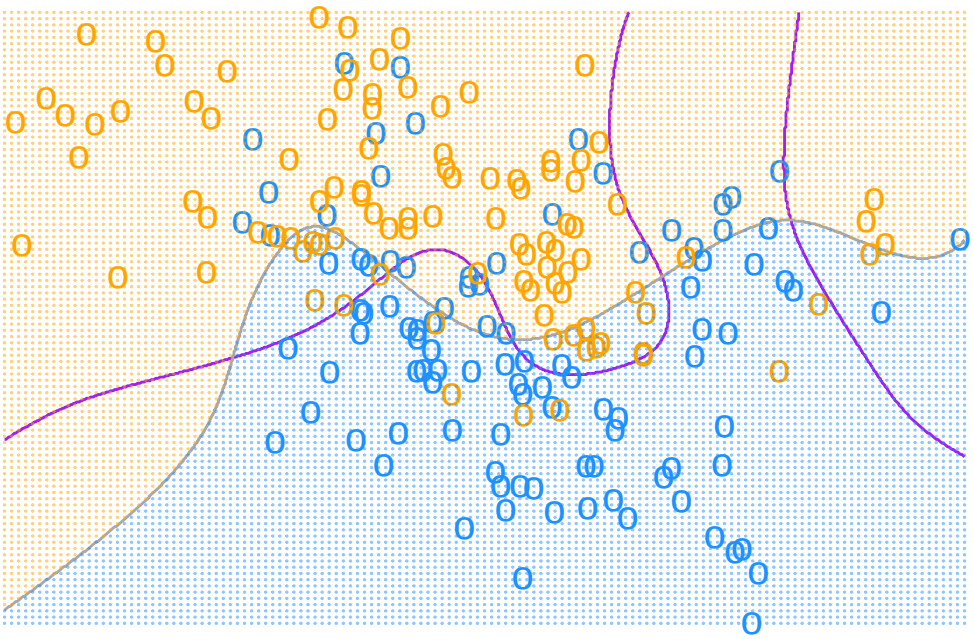
- Degree 3:



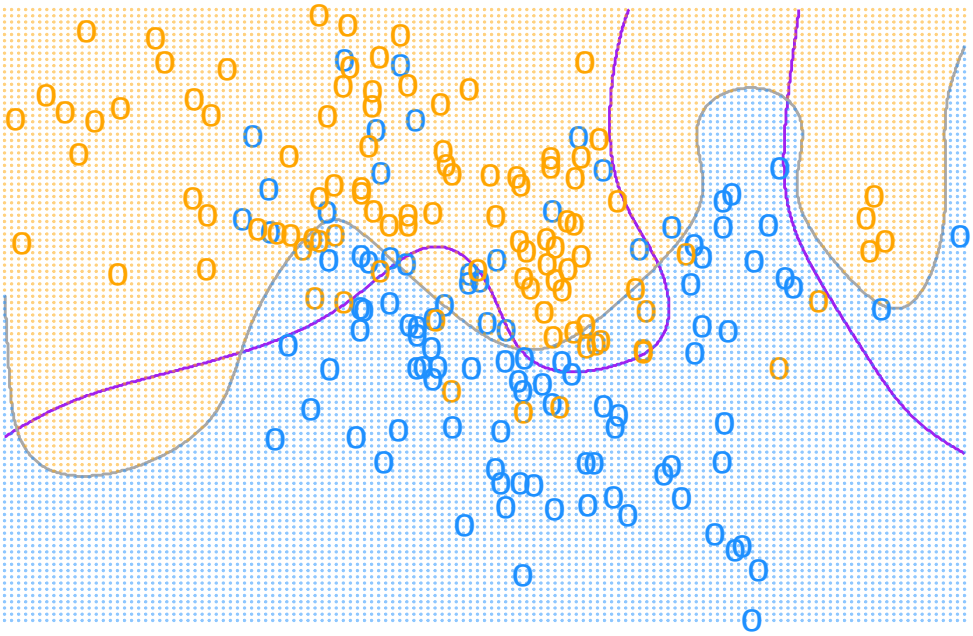
- Degree 4:



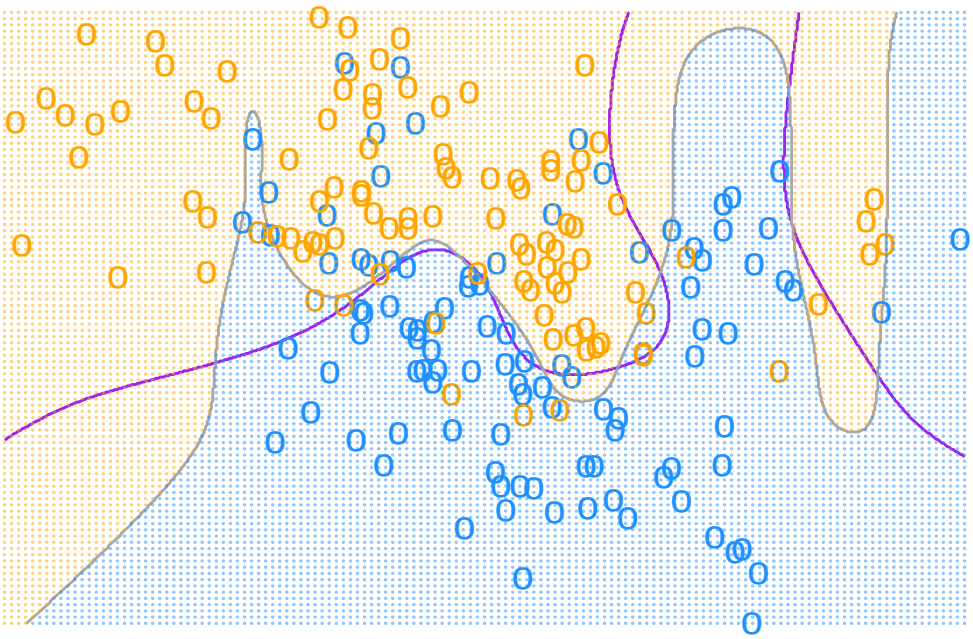
- Degree 5:



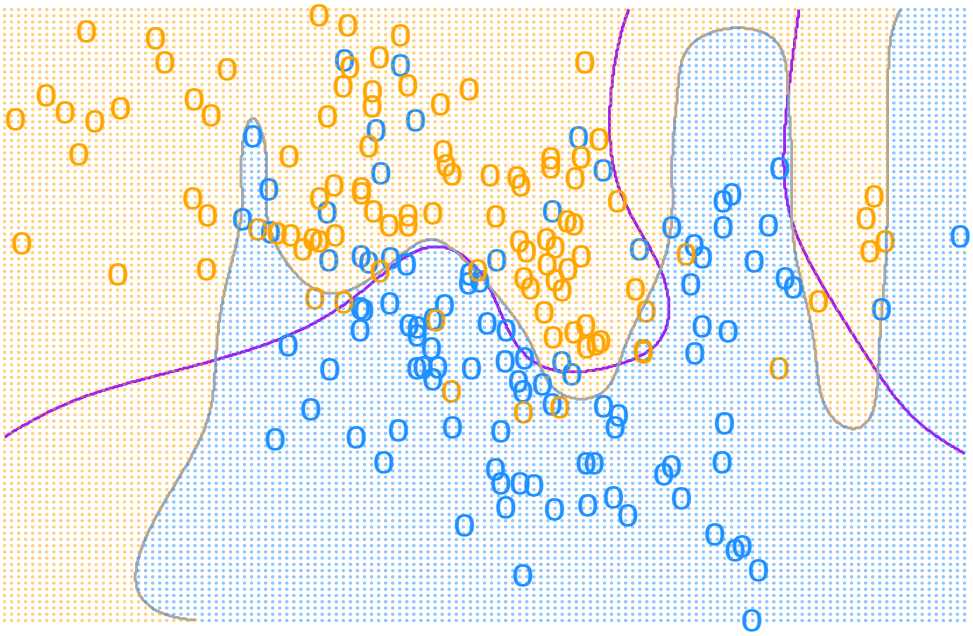
- Degree 6:



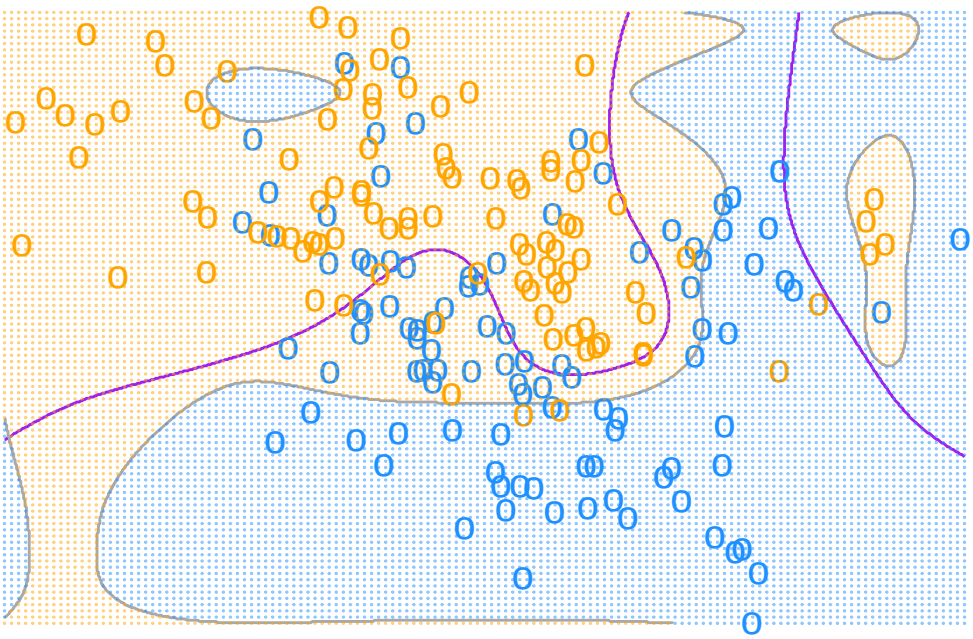
- Degree 7:



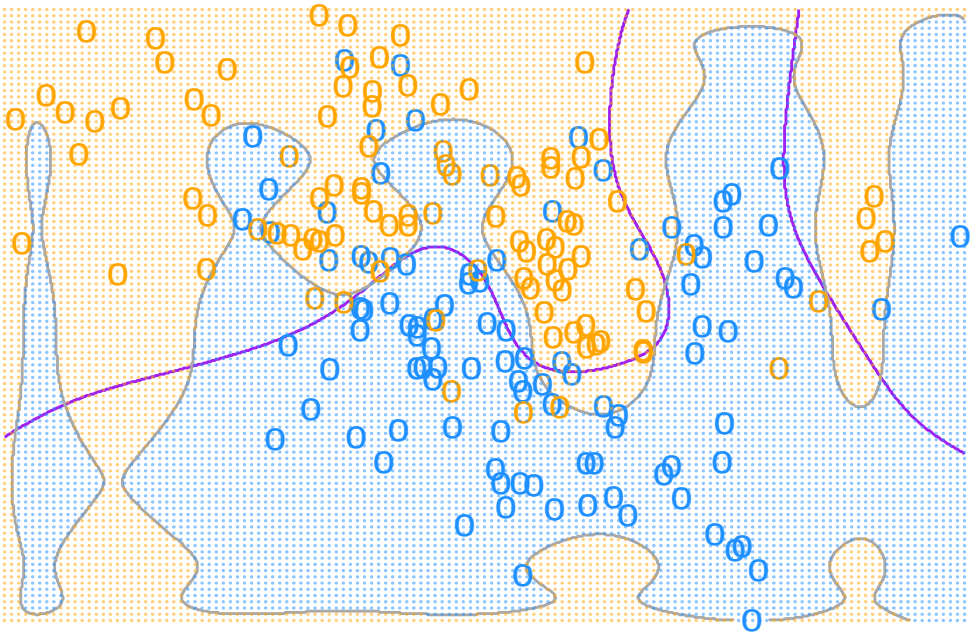
- Degree 8:



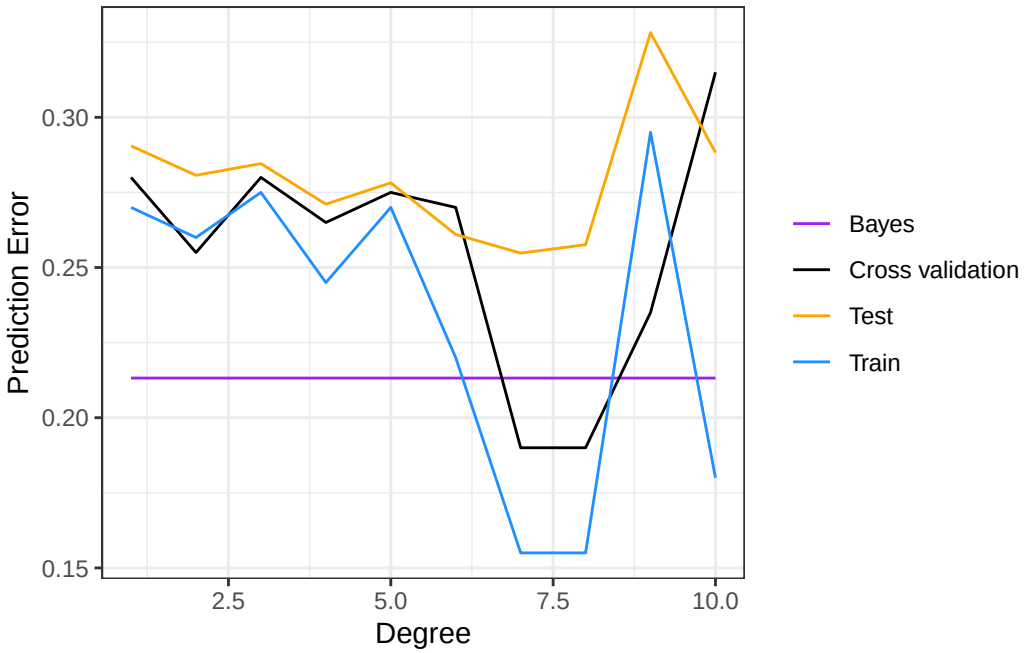
- Degree 9:



- Degree 10:



We plot below the misclassification curve, comparing the misclassification error estimated on a large testing set using the generating distribution (in orange, testing set generated knowing the oracle) with misclassification estimated using a resampling method on the training set (in black, 10-fold Cross Validation, defined later). In our case, we see that misclassification using a resampling method, although optimistic, behaves roughly like misclassification using large testing set:



The cross-validated prediction error behaves nicer in James et al. (2021), using a comparable Mixture data set, but with a lower Bayes error (13% vs 20% for the original):

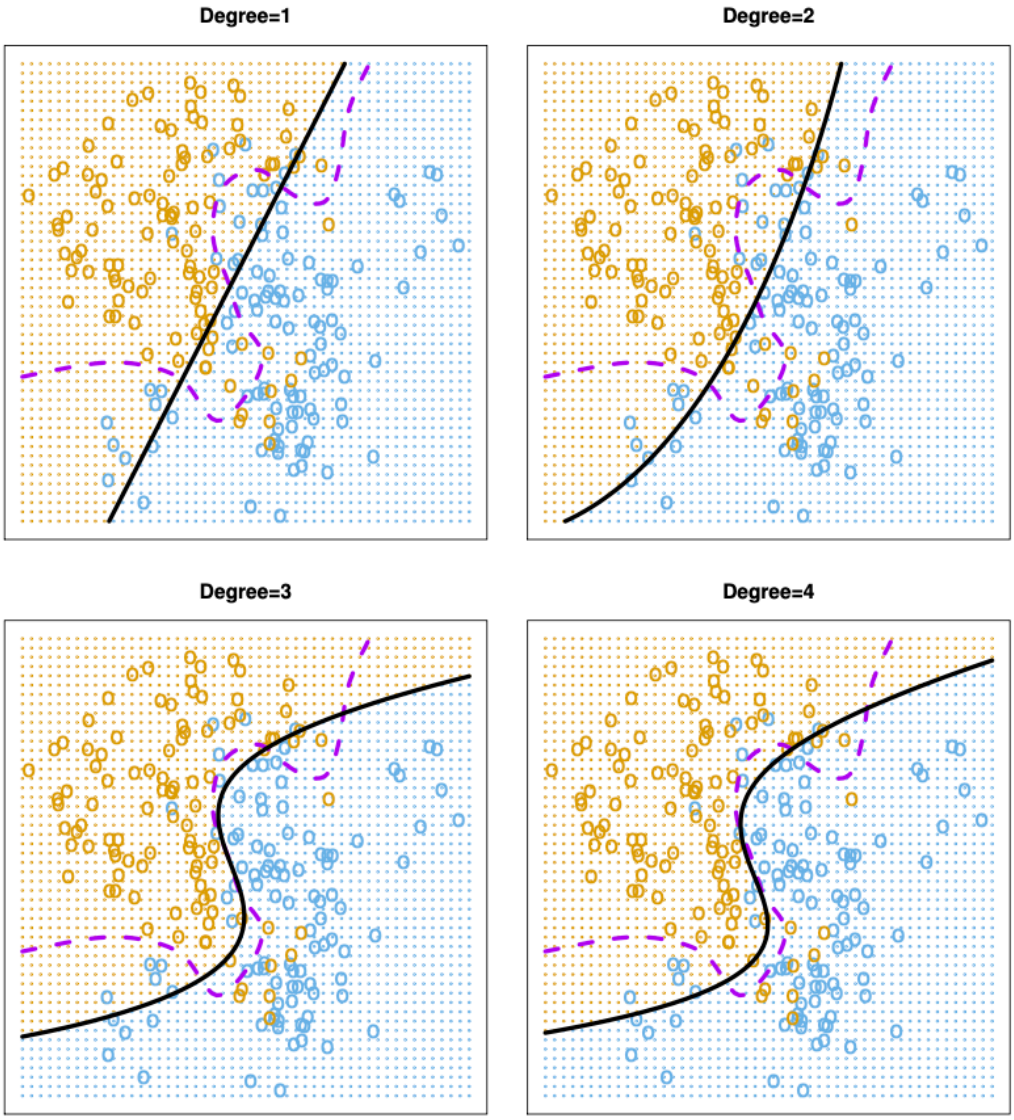


FIGURE 5.7. *Logistic regression fits on the two-dimensional classification data displayed in Figure 2.13. The Bayes decision boundary is represented using a purple dashed line. Estimated decision boundaries from linear, quadratic, cubic and quartic (degrees 1–4) logistic regressions are displayed in black. The test error rates for the four logistic regression fits are respectively* 0.201, 0.197, 0.160, *and* 0.162, *while the Bayes error rate is* 0.133.

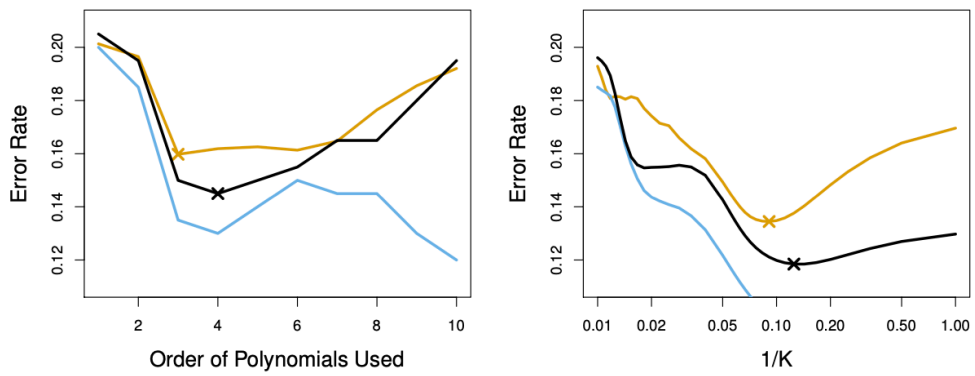


FIGURE 5.8. *Test error (brown), training error (blue), and* 10*-fold CV error (black) on the two-dimensional classification data displayed in Figure 5.7.* Left: *Logistic regression using polynomial functions of the predictors. The order of the polynomials used is displayed on the x-axis.* Right: *The KNN classifier with different values of K, the number of neighbors used in the KNN classifier.*

This intuition advocates the use of resampling methods to assess prediction error in a real life setting (generating distribution unknown, limited testing set), two principal approaches are used: the Hold-out approach and the K-fold Cross-validation approach.

8.4.2 Hold-out approach

It consists in splitting the data set into:

- a learning or training set used to train the classifier or the Score ;
- a validation or test set used to estimate the empirical risk of the classifier or any other metric (ROC curve, AUC).

Algorithm: Hold-out approach

---

1. Using a partition training/validation $\{\mathcal{T}, \mathcal{V}\}$ of the data set $\mathcal{D}$:
   (a) Fit the classifiers $f_1 \cdots, f_m$ on $\mathcal{T}$

(b) Compute the empirical risk $\hat{\mathrm{R}}(f) = \frac{1}{n_{\mathcal{V}}} \sum_i \ell(y_i, f_m(x_i))$ or any other metric on the validation set $\mathcal{V}$

2. Select a single best model f_{m^*} with respect to the empirical risk or metric

---

The main drawback using a single split of data is the possible variability of empirical risk, which is more pregnant when the data set size reduces. Against this issue an alternative approach is to repeat this process on multiple splits of the data.

8.4.3 K-fold Cross-Validation approach

It consists in splitting randomly the data set into K blocks of folds then repeating K times the Hold-out approach, each time using a different block as validation set.

Algorithm: K-fold Cross-Validation approach

---

1. Using a random partition in K blocks $\{\mathcal{D}_1, \cdots, \mathcal{D}_K\}$ of the data set $\mathcal{D}$
2. For $k = 1, \cdots, K$
   (a) $\{\mathcal{T}_k, \mathcal{V}_k\}$ with $\mathcal{V}_k = \mathcal{D}_k$ and $\mathcal{T}_k = \mathcal{D} \setminus \mathcal{D}_k$
   (b) Fit the classifiers $f_1 \cdots, f_m$ on $\mathcal{T}_k$
   (c) Compute the empirical risk $\hat{\mathrm{R}}(f) = \frac{1}{n_{\mathcal{V}_k}} \sum_i \ell(y_i, f_m(x_i))$ or any other metric on the validation set $\mathcal{V}_k$
3. Select a single best model f_{m^*} using the average empirical risk or metric

---

A variant is to first split the data set into a training and validation set. Perform K-fold Cross-Validation on the training set, for example to select some classifier hyper parameter (number of variables or model specification in logistic regression, penalty, etc), then assess the best "optimized" models on the validation set.

Another variant is to repeat the K-fold CV 5 or 10 times to improve the accuracy of the estimated performance and provide and estimate on its variability.

This book chapter gives a practical overview on these methods (how to implement it in R) and also gives references discussing the validity and limitations of the methods.

9 Case study

The article from Desbois (2008) (available here together with a sample data set) shows a complete case study around the detection of financial risks applicable to farm holdings in France. Exploratory Data Analysis is performed in particular using PCA. Linear Discriminant Analysis and Logistic Regression (plus stepwise variable selection) are used and ROC curves are used to compare the two methods.

The data set uses a format specific to the SPSS software, but is readable from R using the `foreign` package.

Using the Agriculture Farm Lending data set:

9.1 EDA

Explore the data set (the full case study is presented in Desbois (2008)), you might try to reproduce the PCA analysis (using package `FactomineR` for example).

```
# package used to import spss file
library(foreign)

don_desbois <- read.spss("./presentation_data/Agriculture Farm Lending/desbois.sav",
                         to.data.frame = TRUE) %>% as_tibble()
don_desbois <- don_desbois %>%
    mutate(Y = as.factor(if_else(DIFF=='healthy', 0, 1))) %>%
    dplyr::select(-DIFF)

glimpse(don_desbois)
```

```
Rows: 1,260
Columns: 30
$ CNTY    <fct> Eure, Eure, Eure, Eure, Eure, Eure, Eure, Eure, Eure, Eure, Eu~
$ STATUS  <fct> Entreprise individuelle, Entreprise individuelle, Entreprise i~
$ HECTARE <dbl> 166, 101, 138, 166, 137, 107, 100, 69, 176, 177, 108, 123, 87,~
$ ToF     <fct> cereals, cereals, dairy farm, cereals, mixed livestock, cereal~
$ OWNLAND <fct> yes, no, yes, yes, yes, yes, no, yes, yes, yes, yes, yes, yes,~
$ AGE     <dbl> 35, 35, 42, 50, 33, 45, 34, 30, 44, 39, 40, 40, 40, 52, 49, 44~
$ HARVEST <fct> 1988, 1988, 1988, 1988, 1988, 1988, 1988, 1988, 1988, 1988, 19~
$ r1      <dbl> 0.449, 0.450, 0.332, 0.363, 0.440, 0.306, 0.717, 0.566, 0.145,~
$ r2      <dbl> 0.622, 0.617, 0.819, 0.733, 0.650, 0.755, 0.320, 0.465, 0.881,~
$ r3      <dbl> 0.25500, 0.24110, 0.55680, 0.35960, 0.31420, 0.26350, 0.16280,~
$ r4      <dbl> 0.11450, 0.10840, 0.18510, 0.13050, 0.13820, 0.08055, 0.11680,~
```

```
$ r5      <dbl> 0.334, 0.341, 0.147, 0.232, 0.302, 0.225, 0.601, 0.499, 0.115,~
$ r6      <dbl> 0.785, 0.518, 0.700, 0.773, 0.846, 0.709, 0.894, 0.863, 0.439,~
$ r7      <dbl> 0.585, 0.393, 0.310, 0.495, 0.580, 0.523, 0.748, 0.761, 0.348,~
$ r8      <dbl> 0.20020, 0.12500, 0.38950, 0.27790, 0.26580, 0.18700, 0.14550,~
$ r11     <dbl> 0.6628, 0.7098, 0.4142, 0.4661, 0.7715, 0.8178, 0.6598, 0.6619~
$ r12     <dbl> 1.3698, 1.2534, 0.6370, 1.0698, 1.4752, 1.4682, 1.1018, 1.2360~
$ r14     <dbl> 0.23200, 0.14970, 0.48470, 0.37350, 0.25630, 0.18610, 0.18070,~
$ r17     <dbl> 0.0884, 0.0671, 0.0445, 0.0621, 0.0489, 0.0243, 0.0352, 0.0879~
$ r18     <dbl> 0.0694, 0.0348, 0.0311, 0.0480, 0.0414, 0.0173, 0.0315, 0.0758~
$ r19     <dbl> 0.1660, 0.1360, 0.1030, 0.1080, 0.1270, 0.0652, 0.1290, 0.2400~
$ r21     <dbl> 0.1340, 0.0802, 0.0890, 0.0851, 0.0838, 0.0390, 0.0784, 0.1630~
$ r22     <dbl> 0.3219, 0.3133, 0.2945, 0.1909, 0.2567, 0.1472, 0.3211, 0.5170~
$ r24     <dbl> 0.295, 0.365, 0.166, 0.265, 0.257, 0.191, 0.322, 0.305, 0.180,~
$ r28     <dbl> 0.475, 0.434, 0.350, 0.475, 0.475, 0.443, 0.401, 0.464, 0.475,~
$ r30     <dbl> 0.35000, 0.29780, 0.24680, 0.37590, 0.36690, 0.37590, 0.27240,~
$ r32     <dbl> 0.4313, 0.3989, 0.3187, 0.4313, 0.4313, 0.4257, 0.3697, 0.3886~
$ r36     <dbl> 0.886, 0.351, 1.300, 1.385, 0.886, 1.316, 0.441, 0.761, 2.147,~
$ r37     <dbl> 0.572, 0.867, 0.475, 0.470, 0.520, 0.431, 0.803, 0.656, 0.359,~
$ Y       <fct> 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,~
```

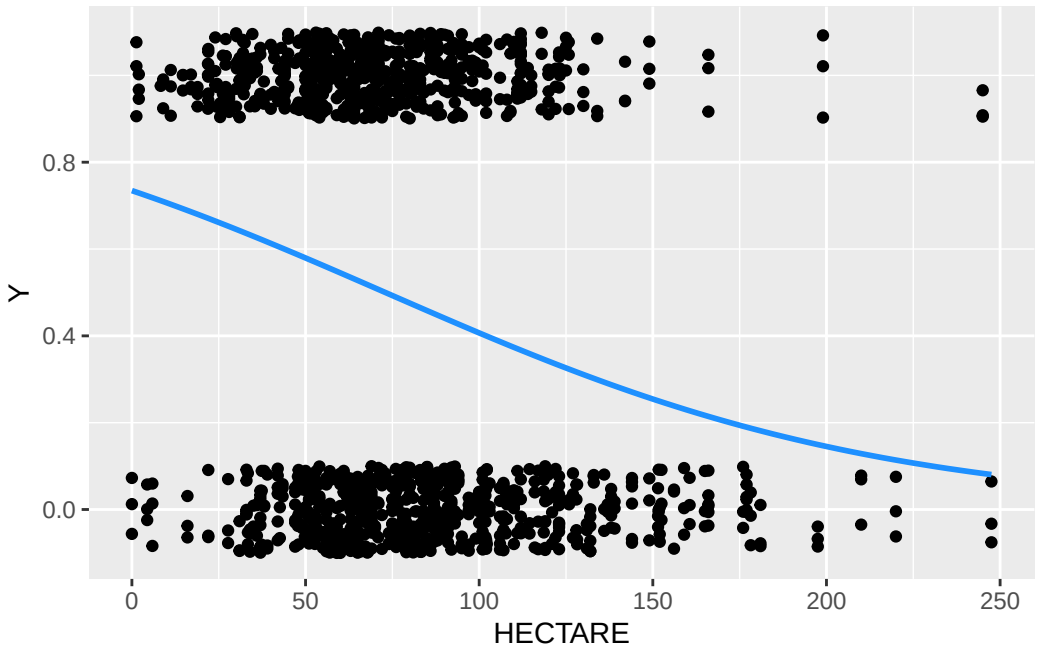
```
# Variable definitions from Desbois article
# Capitalization
# r1 total debt / total assets;
# r2 stockholders' equity / invested capital;
# r3 short term debt / total debt;
# r4 short term debt / total assets;
# r5 long and medium term debt / total assets;

# Weight of the debt
# r6 total debt / gross product;
# r7 long and medium term debt / gross product;
# r8 short term debt / gross product;

# Liquidity
# r11 working capital / gross product;
# r12 working capital / (real inputs - financial expenses);
# r14 short term debt / circulating assets;

# Debt servicing
# r17 financial expenses / total debt;
# r18 financial expenses / gross product;
# r19 (financial expenses+ refunding of long and
```

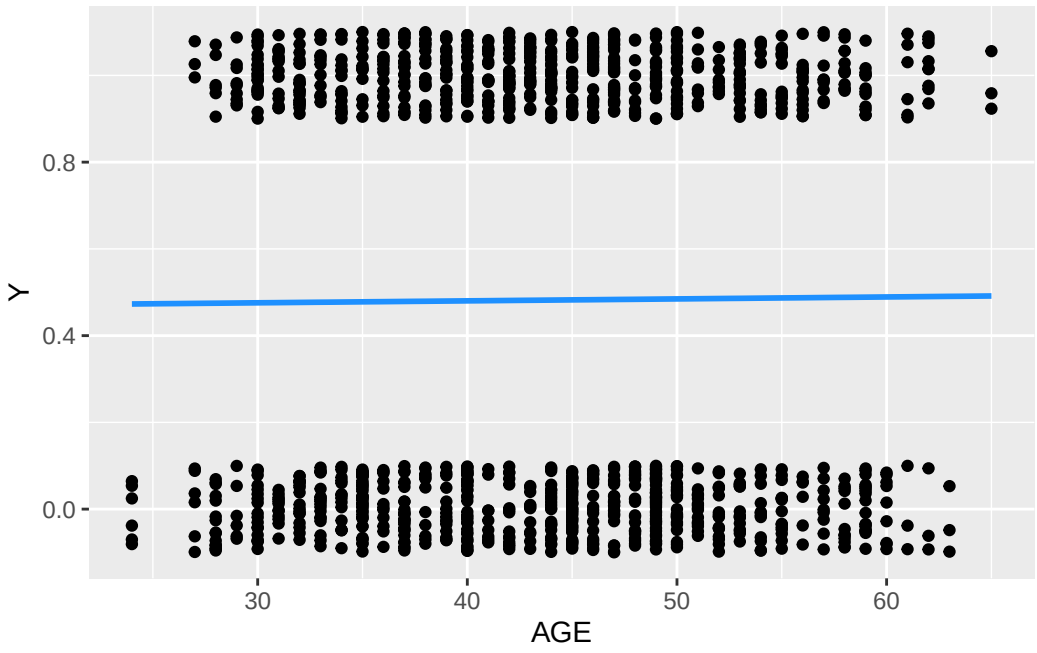
```
#      medium term capital) / gross product;
# r21 financial expenses / EBITDA;
# r22 (financial expenses + refunding of long and
#      medium term capital)/EBITDA;

# Capital profitability
# r24 EBITDA /total assets;

# Earnings
# r28 EBITDA / gross product;
# r30 available income / gross product;
# r32 (EBITDA - financial expenses) / gross product;

# Productive activity
# r36 immobilized assets / gross product;
# r37 gross product / total assets.
```

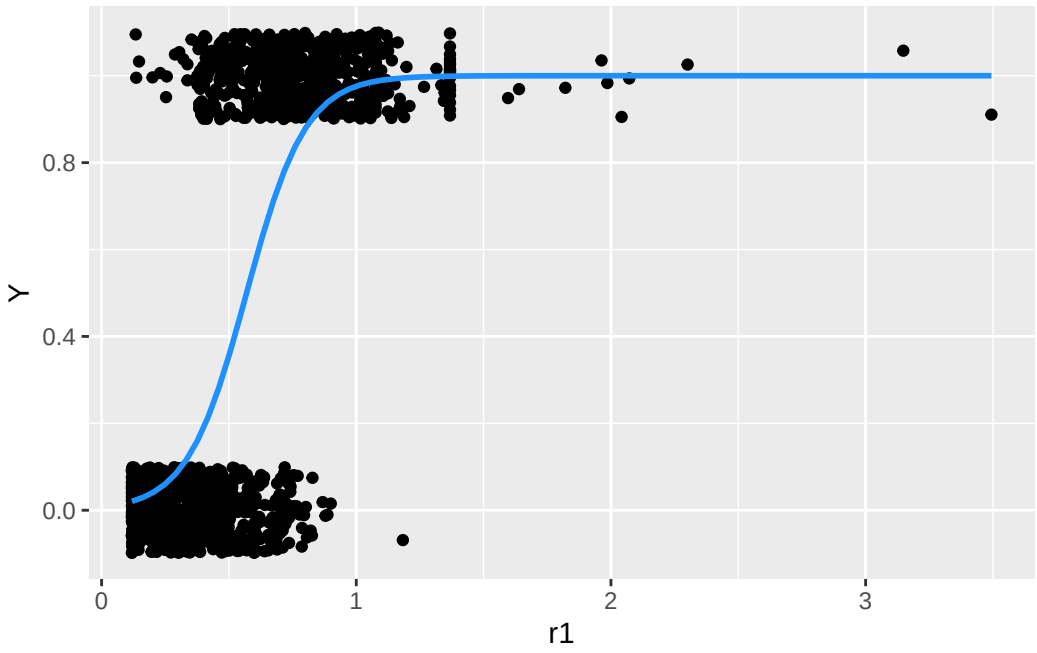
```
# YOUR CODE HERE
```

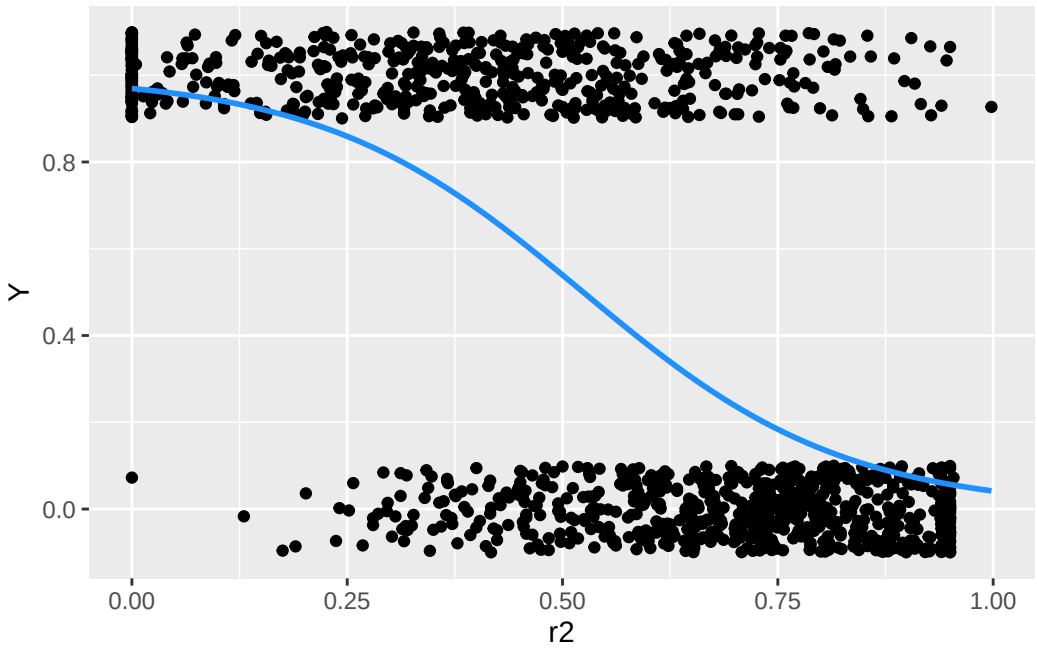
We first plot the bivariate (scatter plots) relationships between the variable of interest and the quantitative predictors:

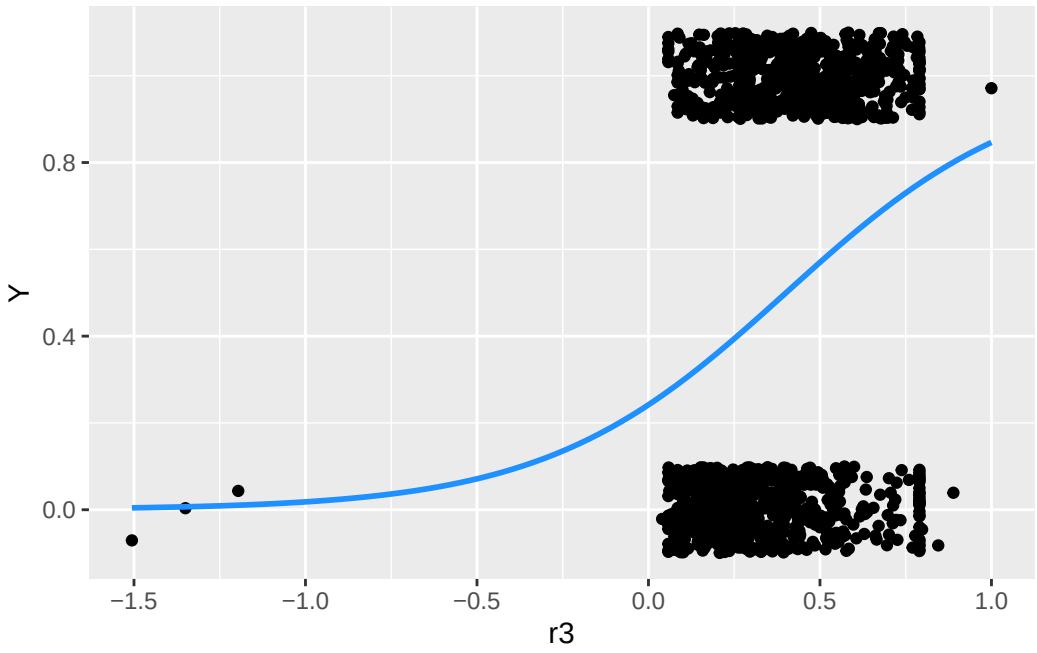
```
vars_quanti <- names(don_desbois %>% select_if(is.numeric))
for(var in vars_quanti){
    var <- as.name(var)
    print(ggplot(don_desbois %>%
                     # filter(r17 <0.14) %>%
                     mutate(Y = as.numeric(Y)-1), aes(x=!! var,y = Y)) +
        geom_jitter(height = 0.1, width = 0) +
        geom_smooth(method = "glm",
                    formula = y ~ x,
                    method.args = list(family = "binomial"),
                    se = FALSE,
                    col = "dodgerblue"))
}
```

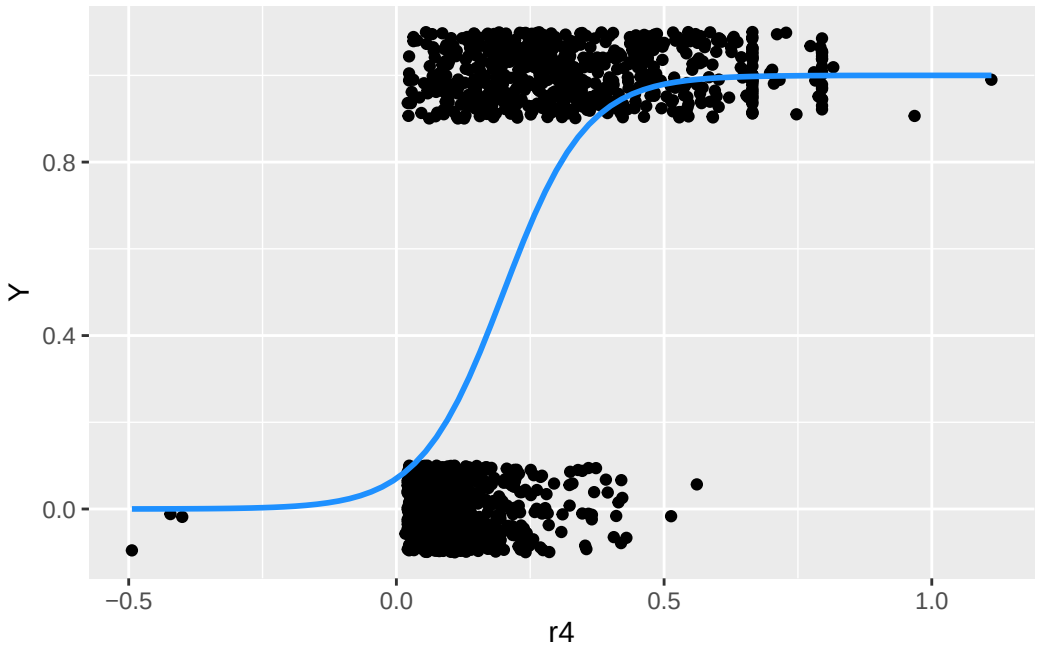

0.8
0.4
0.0
Y
0
50
100
150
200
250
HECTARE

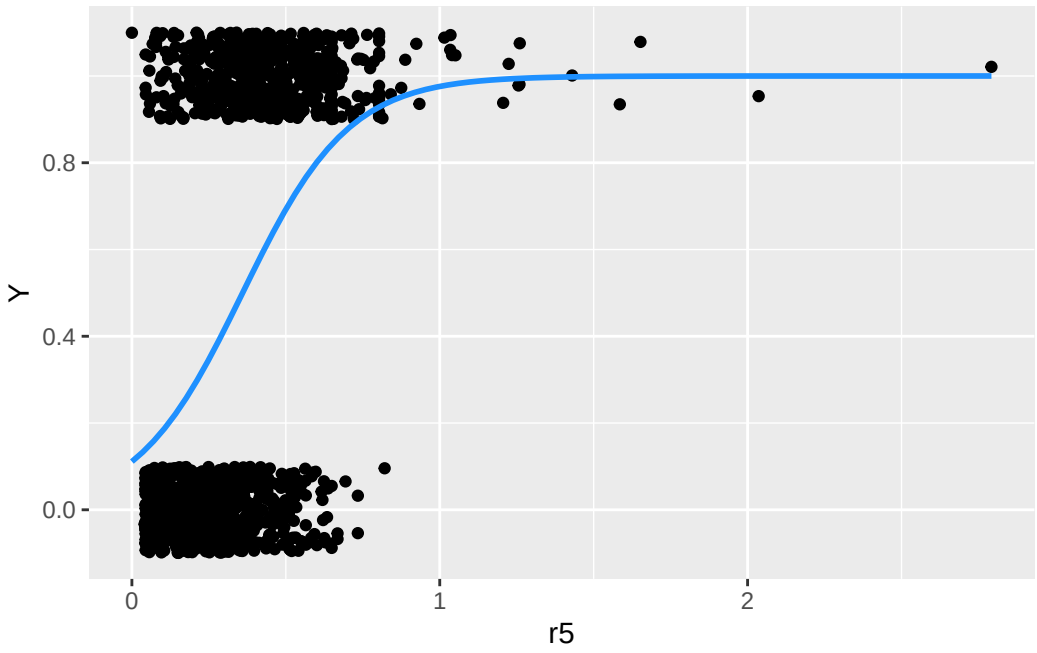

0.8
0.4
0.0
Y
30
40
50
60
AGE

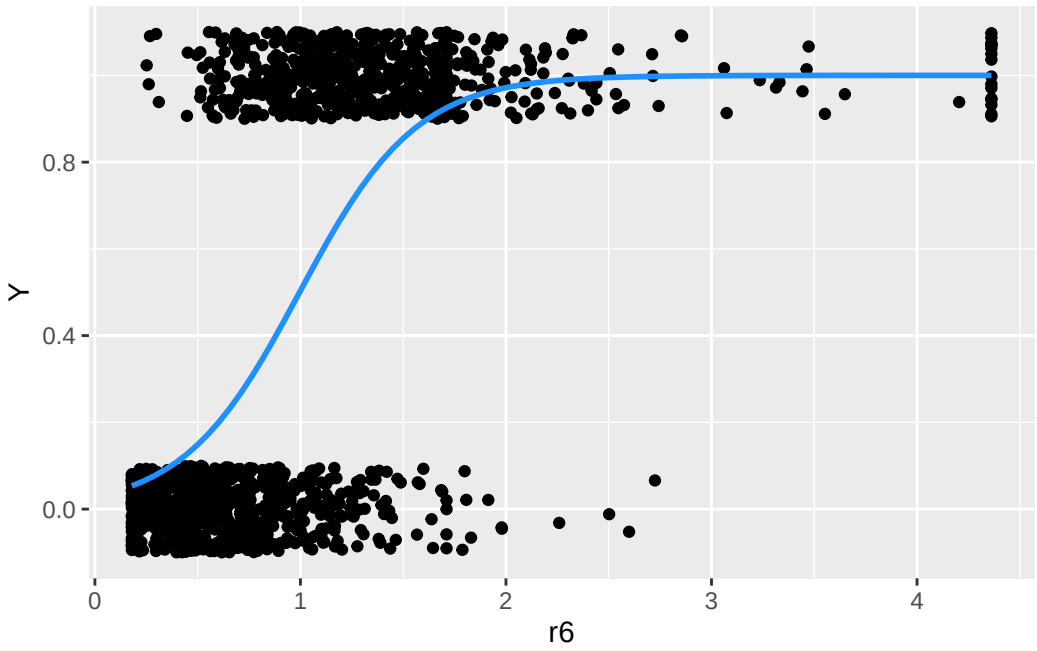

0.8
0.4
0.0
Y
0
1
2
3
r1

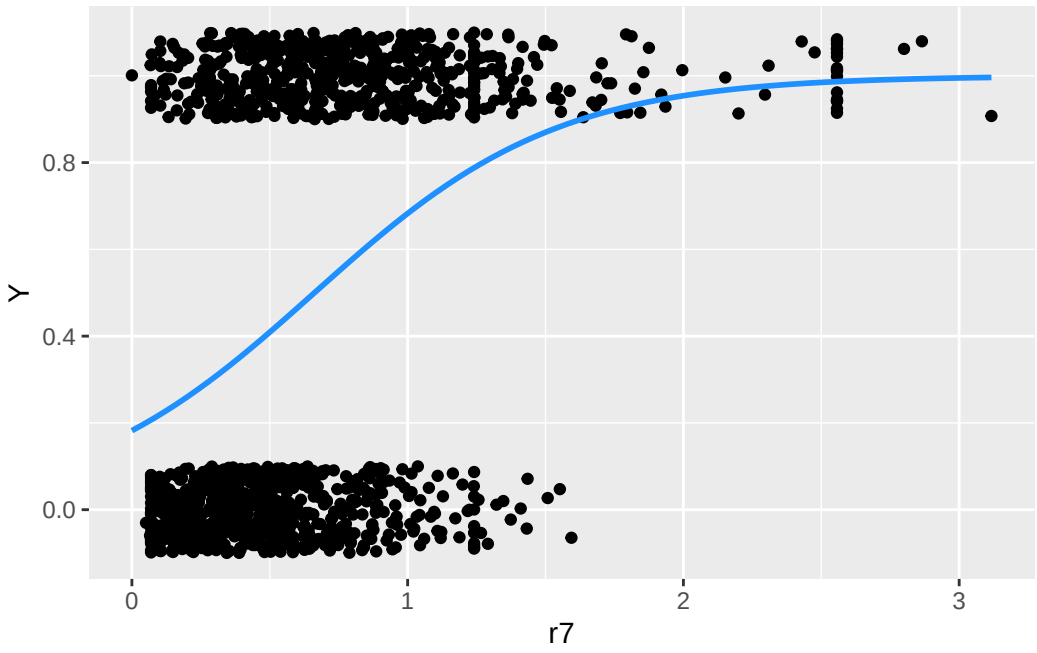

0.8
0.4
0.0
Y
0.00
0.25
0.50
0.75
1.00
r2

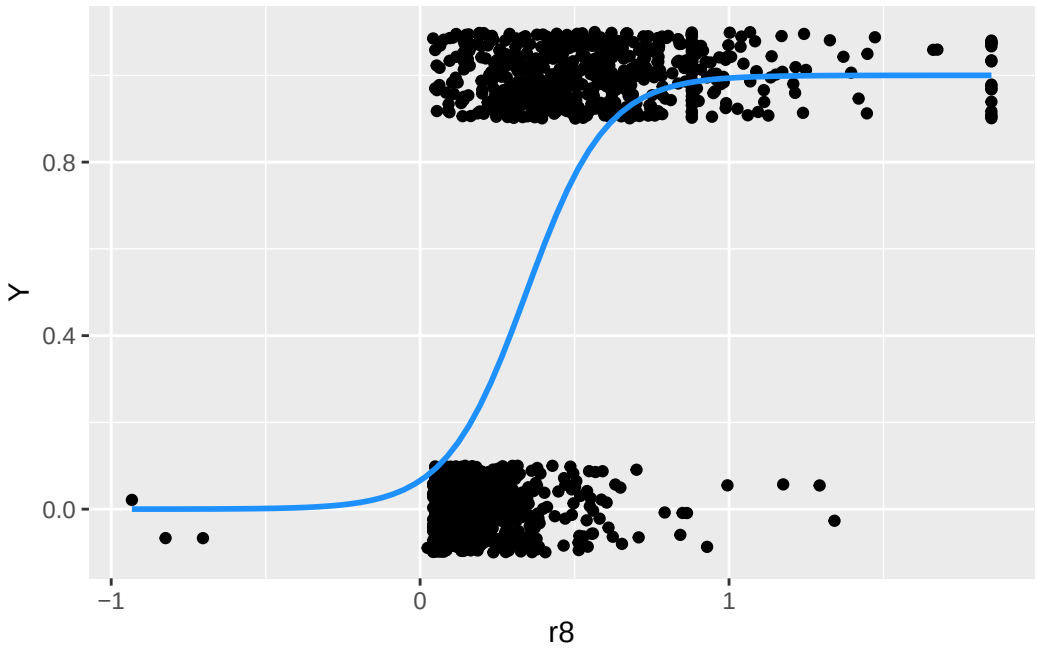

0.8
0.4
0.0
Y
−1.5
−1.0
−0.5
0.0
0.5
1.0
r3

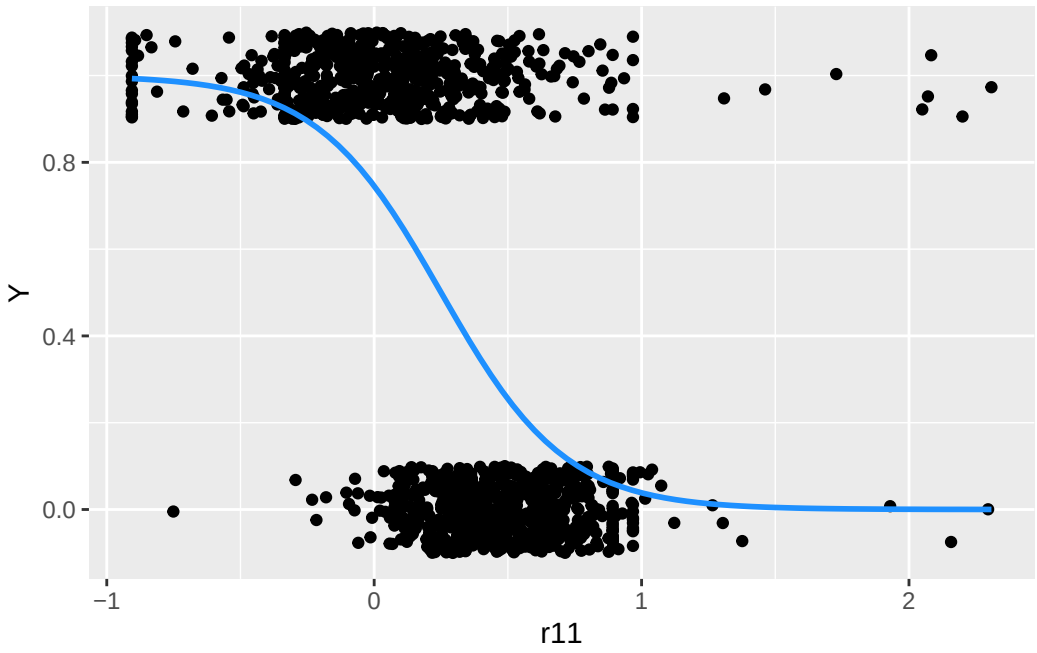

0.8
0.4
0.0
Y
−0.5
0.0
0.5
1.0
r4

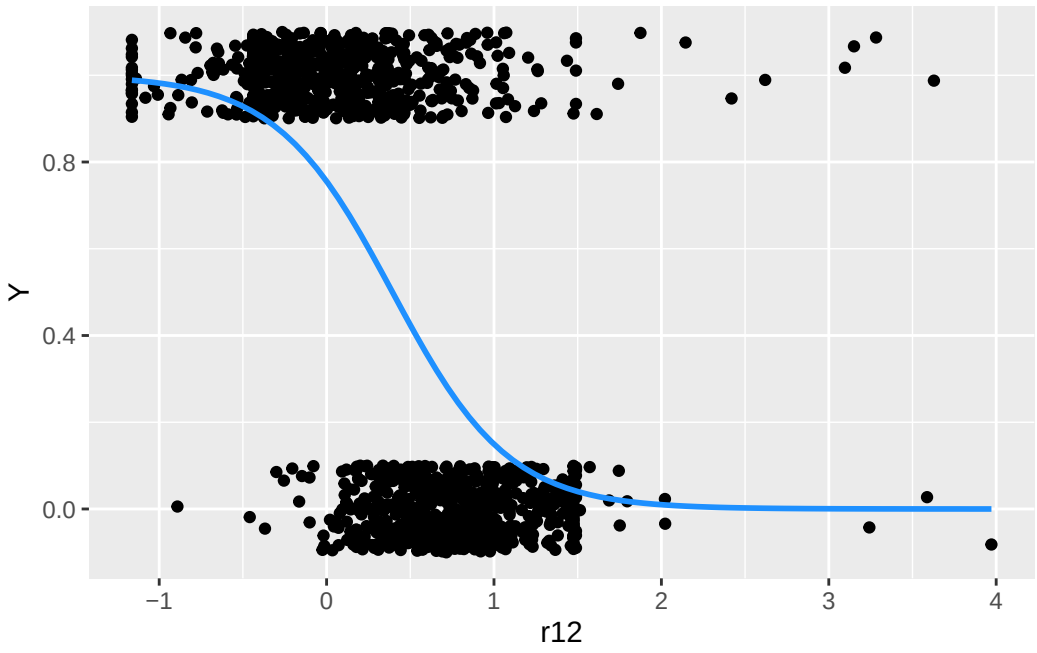

0.8
0.4
0.0
Y
0
1
2
r5

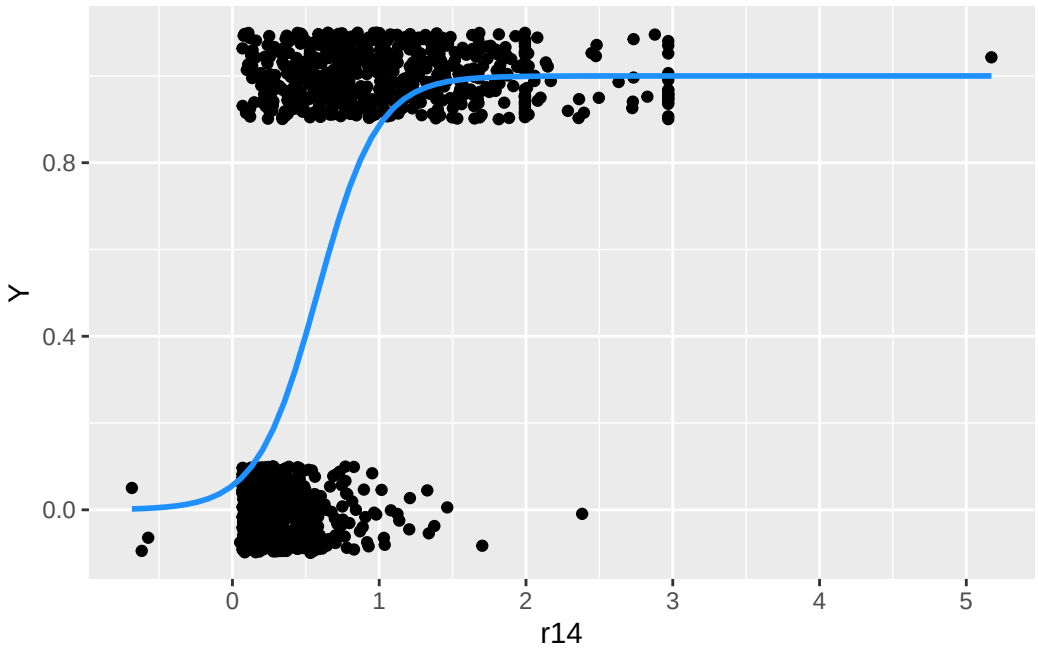

0.8
0.4
0.0
Y
0
1
2
3
4
r6

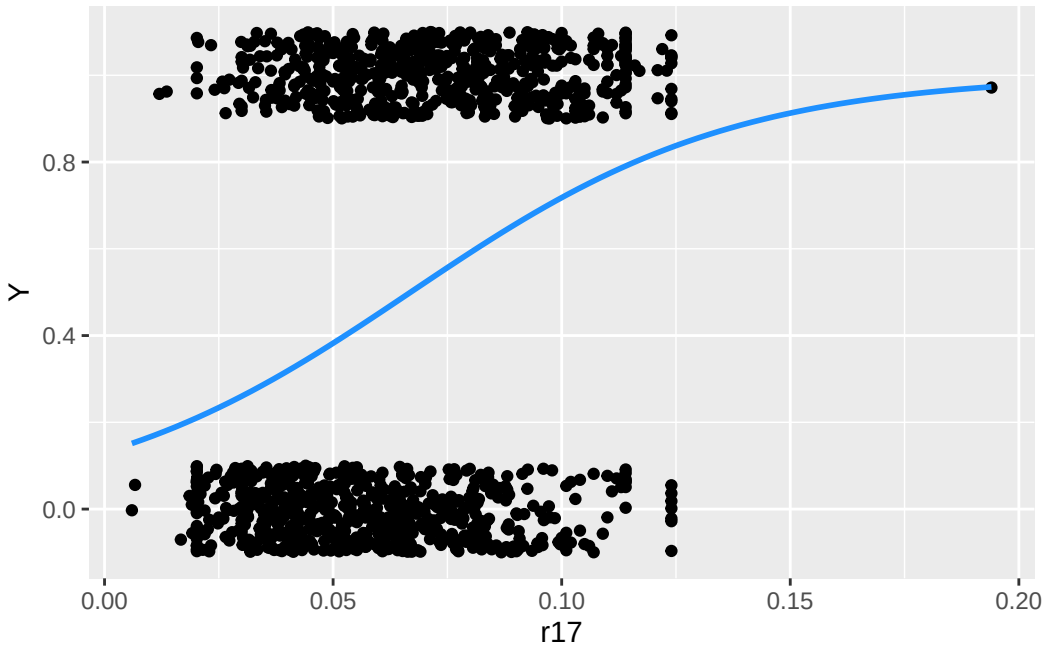

0.8
0.4
0.0
Y
0
1
2
3
r7

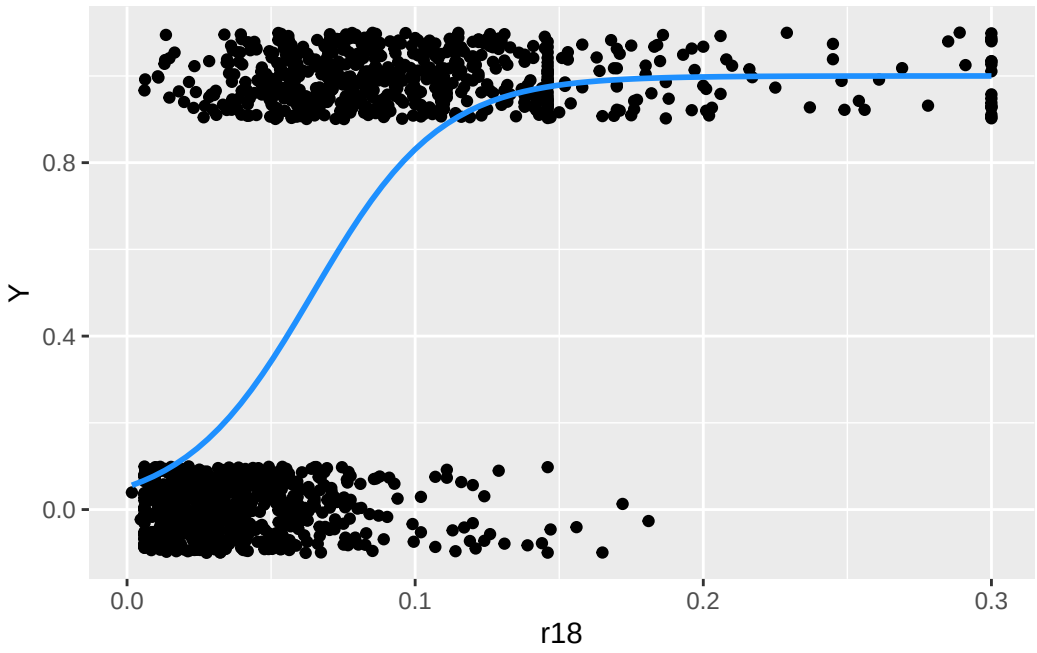

0.8
0.4
0.0
Y
−1
0
1
r8

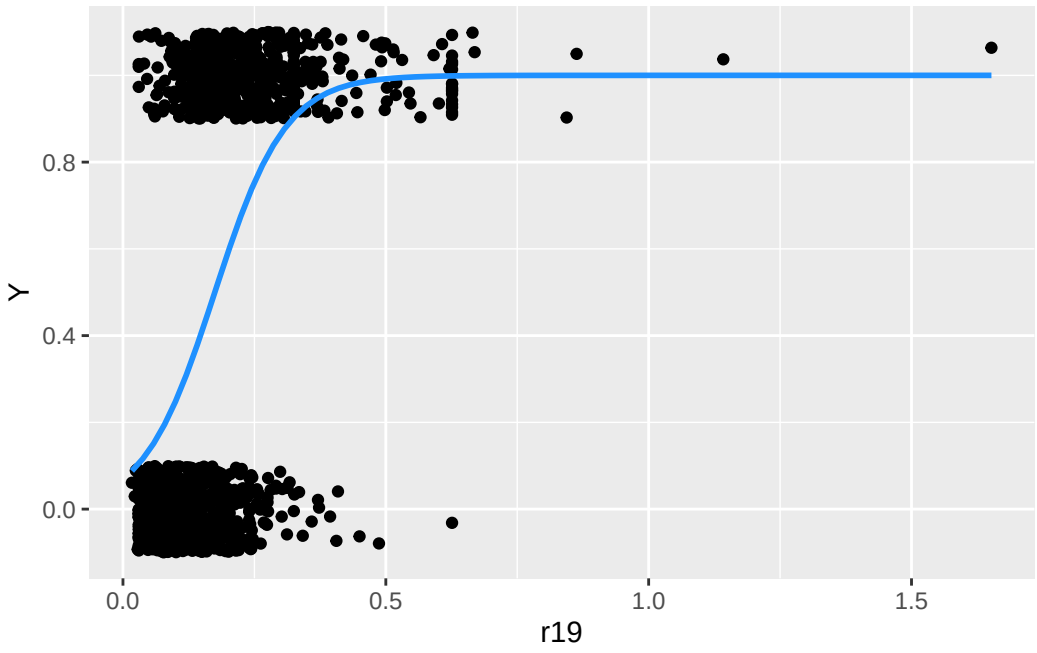

0.8
0.4
0.0
Y
−1
0
1
2
r11

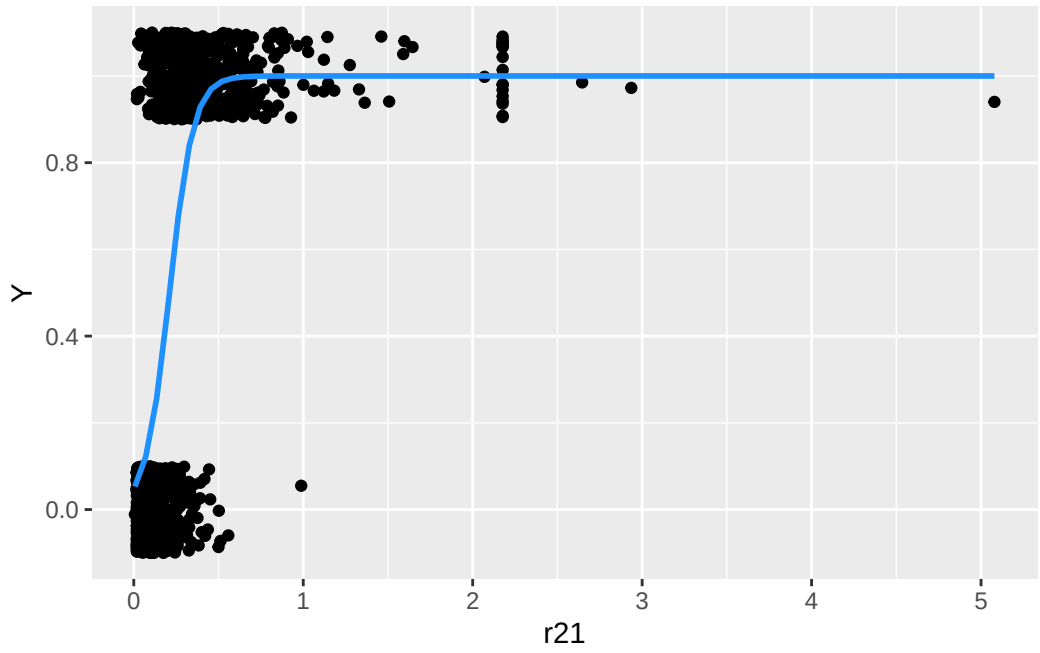

0.8
0.4
0.0
Y
−1
0
1
2
3
4
r12

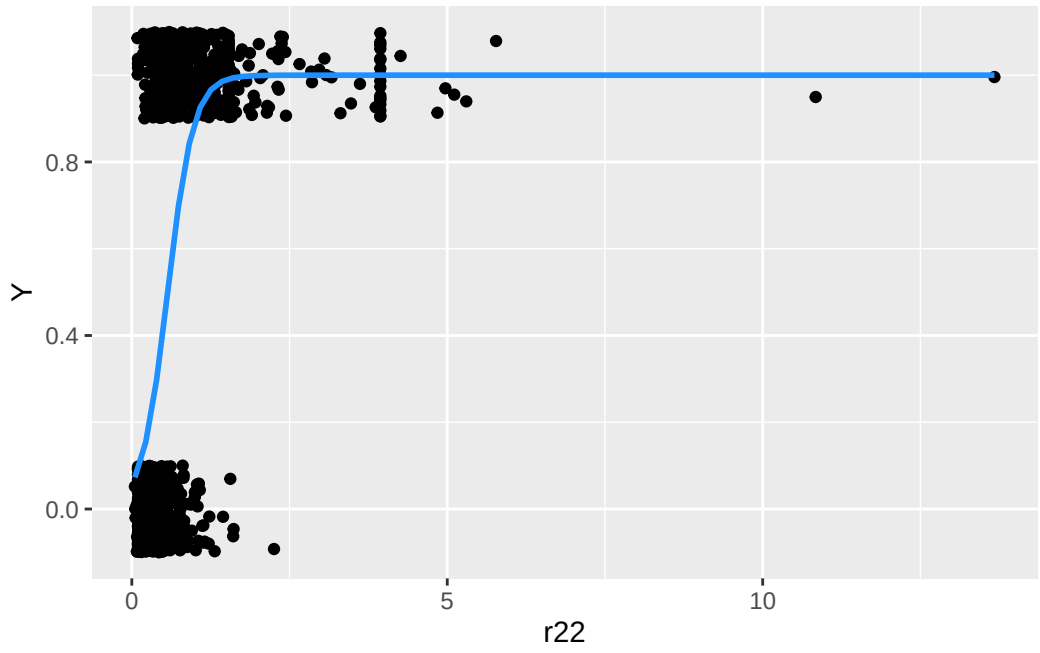

0.8
0.4
0.0
Y
0
1
2
3
4
5
r14

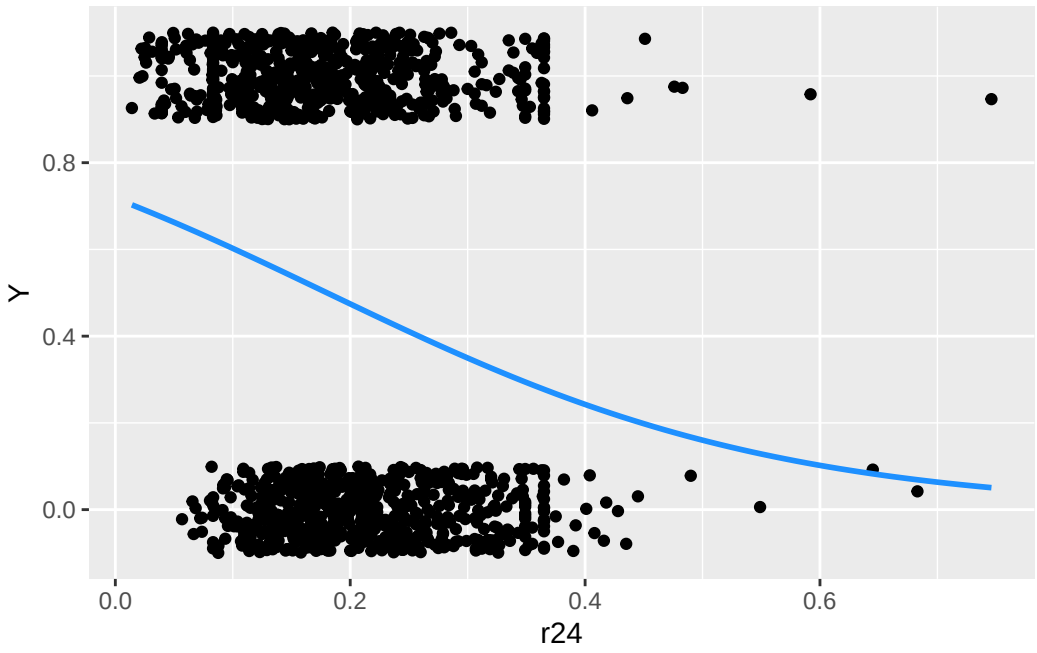

0.8
0.4
0.0
Y
0.00
0.05
0.10
0.15
0.20
r17

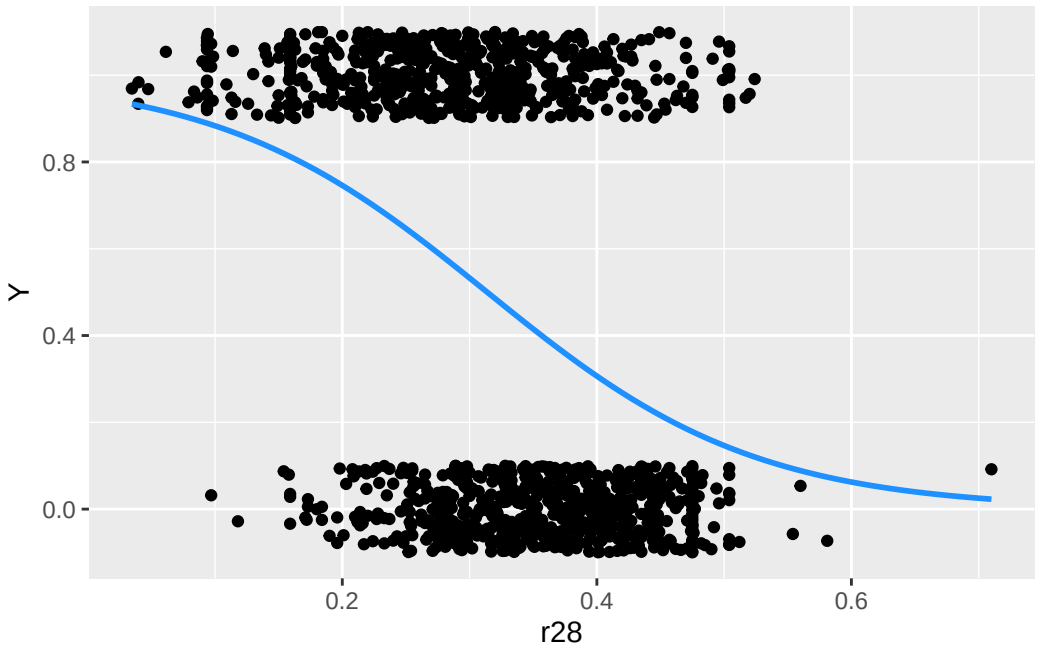

0.8
0.4
0.0
Y
0.0
0.1
0.2
0.3
r18

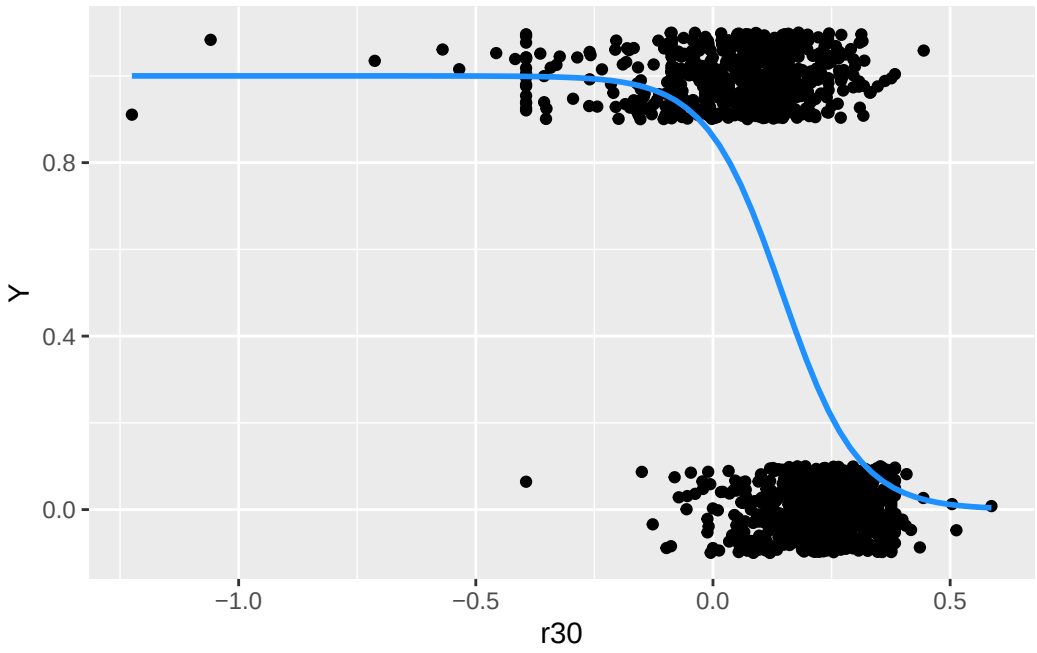

0.8
0.4
0.0
Y
0.0
0.5
1.0
1.5
r19

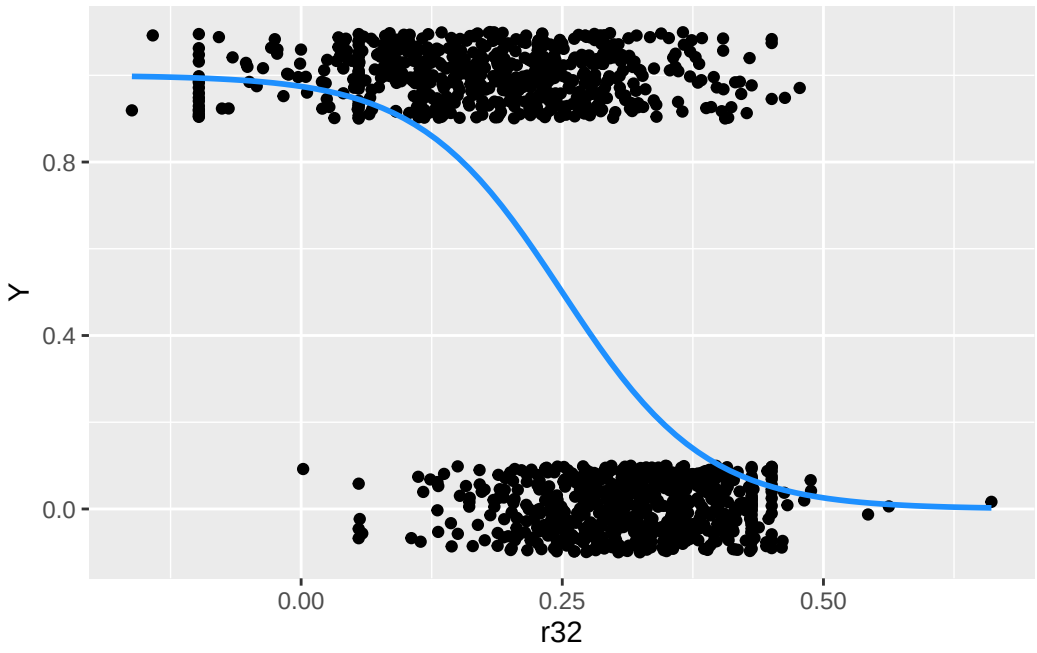

0.8
0.4
0.0
Y
0
1
2
3
4
5
r21

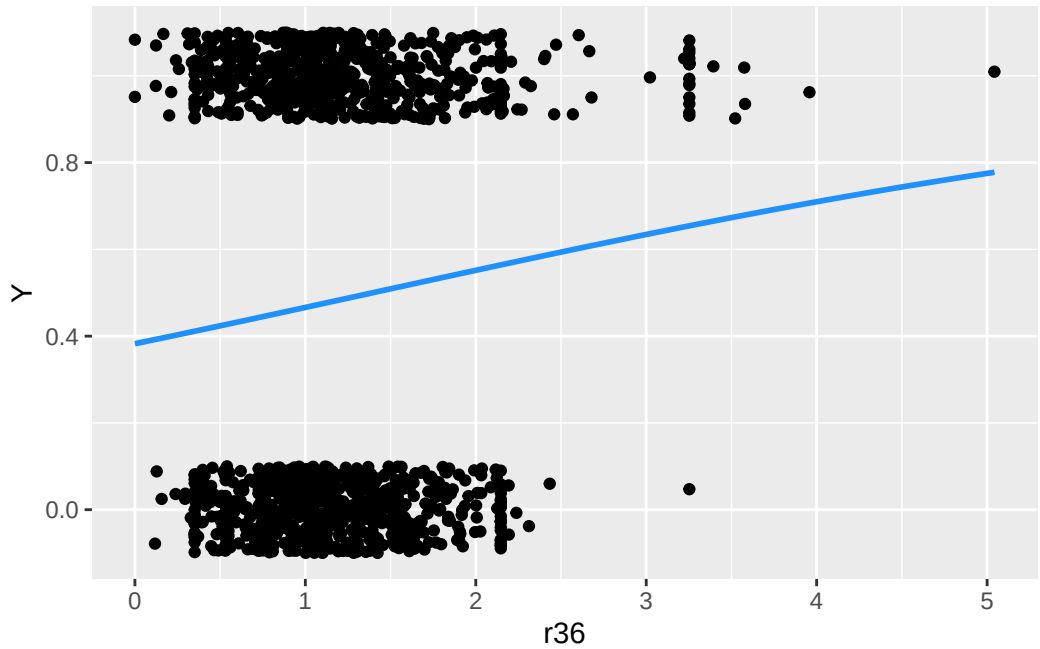

0.8
0.4
0.0
Y
0
5
10
r22

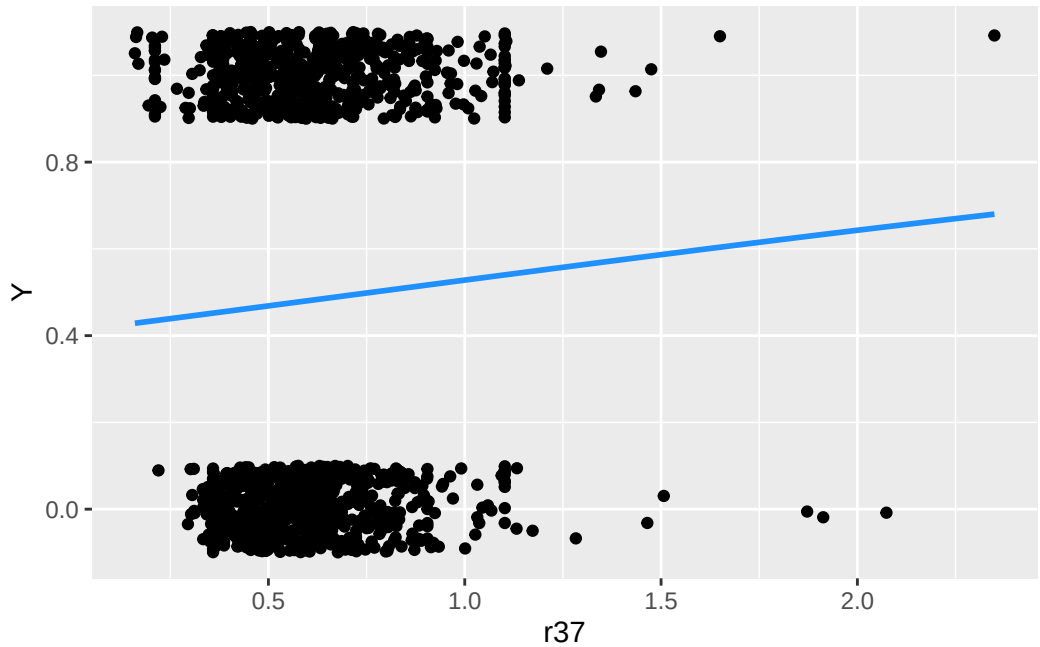

0.8
0.4
0.0
Y
0.0
0.2
0.4
0.6
r24


0.8
0.4
0.0
Y
0.2
0.4
0.6
r28


0.8
0.4
0.0
Y
−1.0
−0.5
0.0
0.5
r30


0.8
0.4
0.0
Y
0.00
0.25
0.50
r32

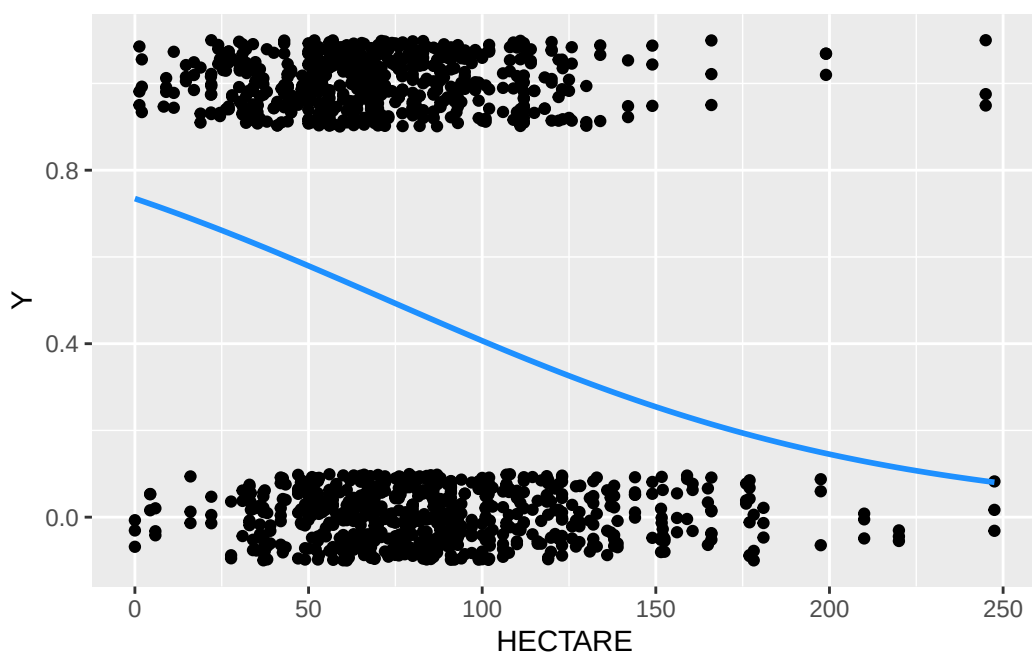


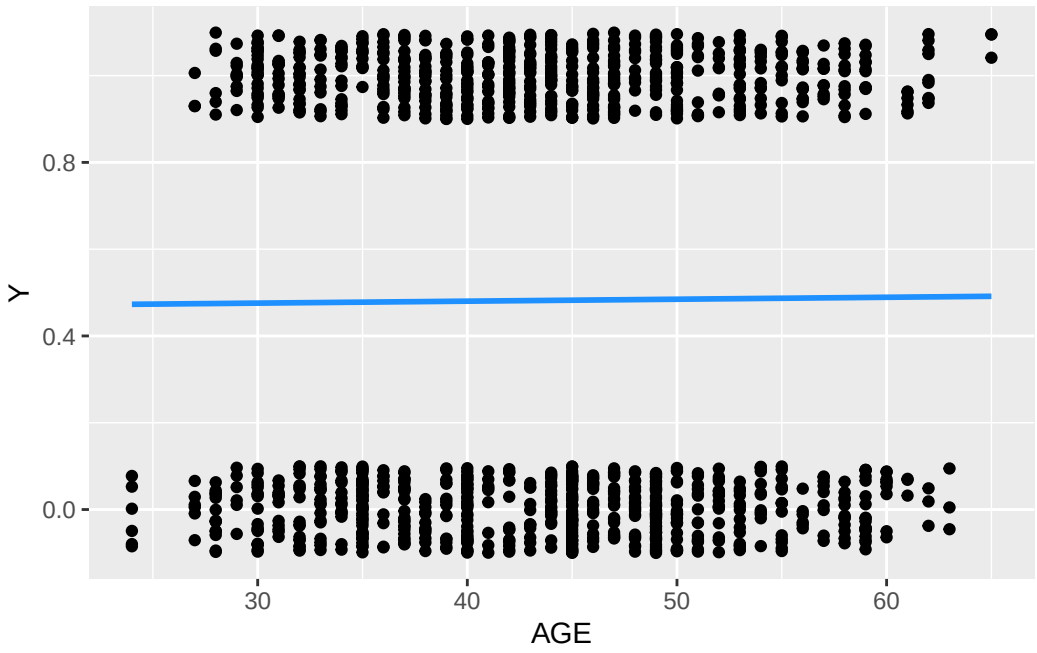


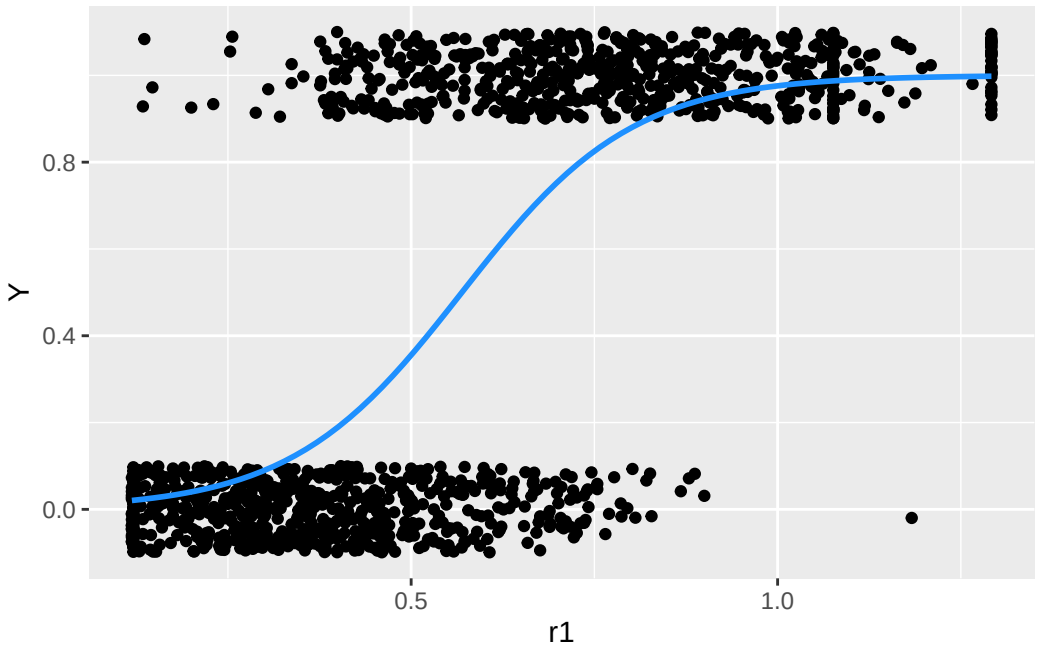
At first sight there seem to be outliers in the financial ratios values (for example r18, r19, r21). We can deal with that for example by winsorizing variables, in real life, if we were responsible

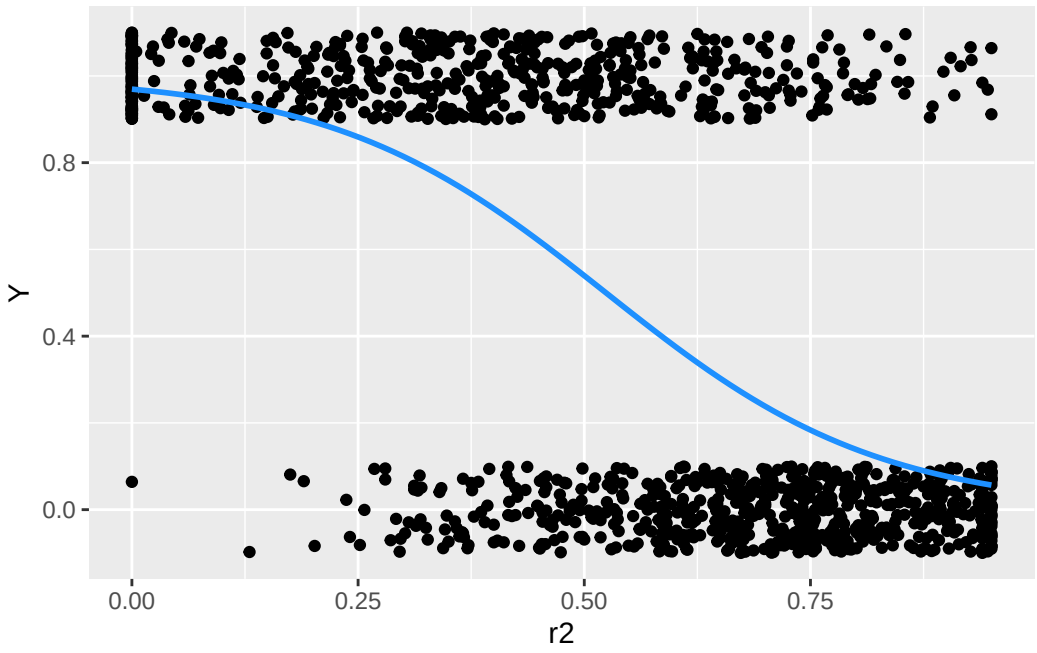
for the data collection, it would be a good idea to investigate closely these outliers.

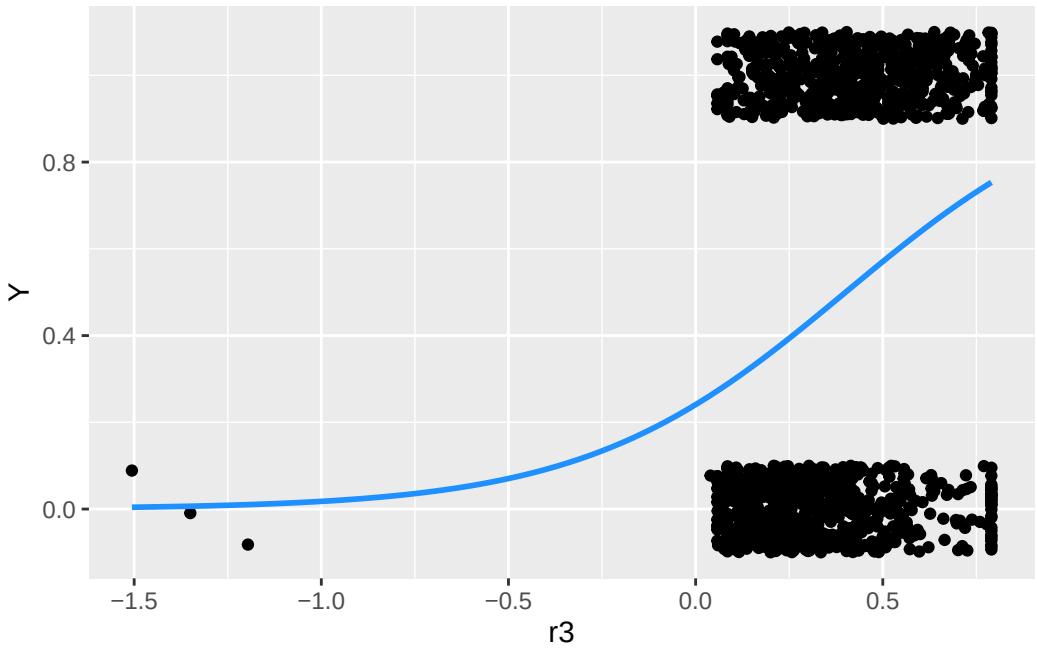
```
don_desbois_winsorized <- don_desbois %>%
    mutate(across(r1:r37, ~ DescTools::Winsorize(.x , quantile( .x, probs = c(0.00, 0.975)

vars_quanti <- names(don_desbois_winsorized %>% select_if(is.numeric))
for(var in vars_quanti){
    var <- as.name(var)
    print(ggplot(don_desbois_winsorized %>%
                     # filter(r17 <0.14) %>%
                     mutate(Y = as.numeric(Y)-1), aes(x=!! var,y = Y)) +
        geom_jitter(height = 0.1, width = 0) +
        geom_smooth(method = "glm",
                    formula = y ~ x,
                    method.args = list(family = "binomial"),
                    se = FALSE,
                    col = "dodgerblue"))
}
```

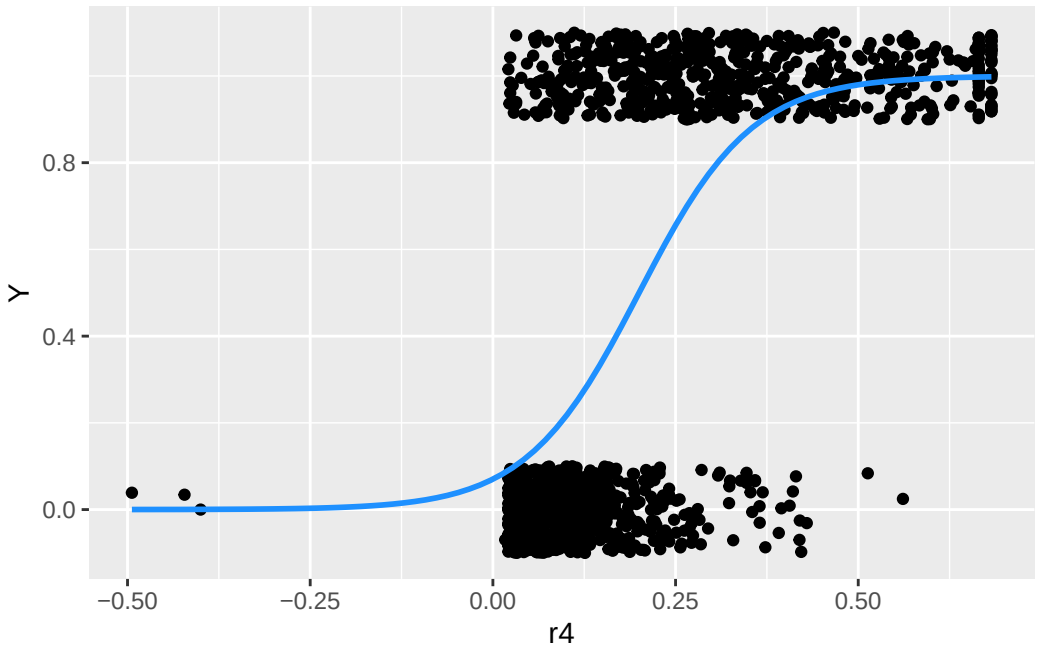


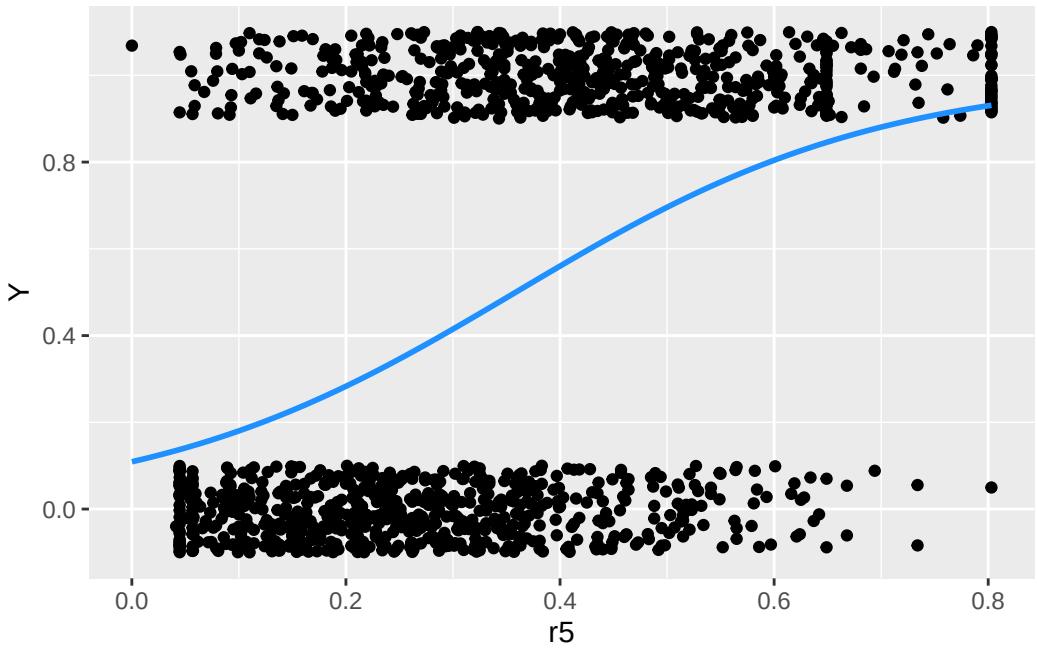

0.8
0.4
0.0
Y
30
40
50
60
AGE

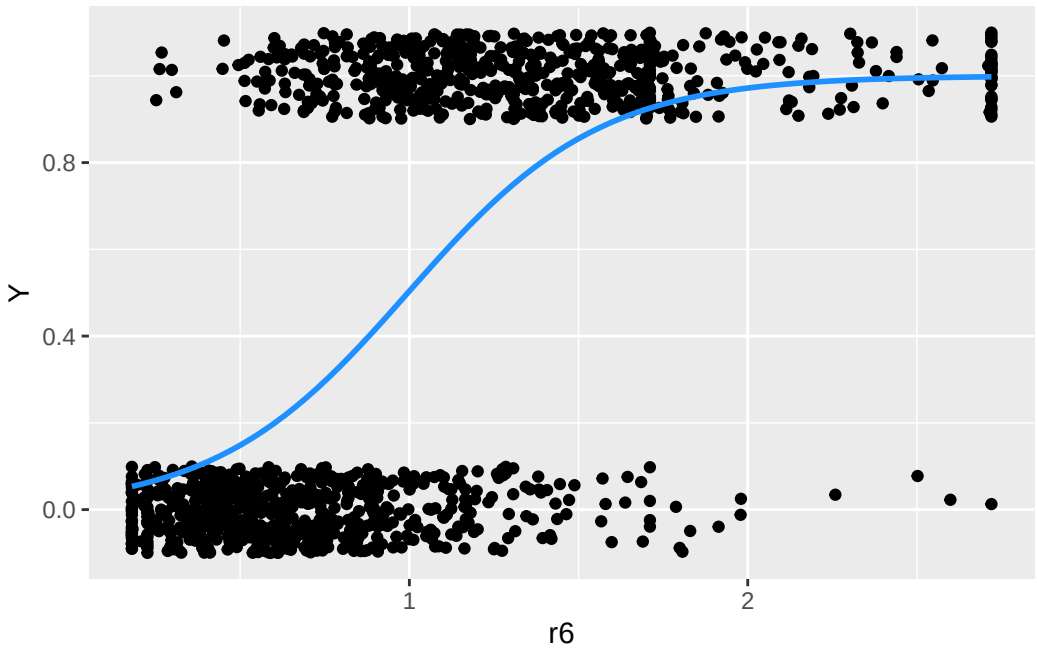

0.8
0.4
0.0
Y
0.5
1.0
r1

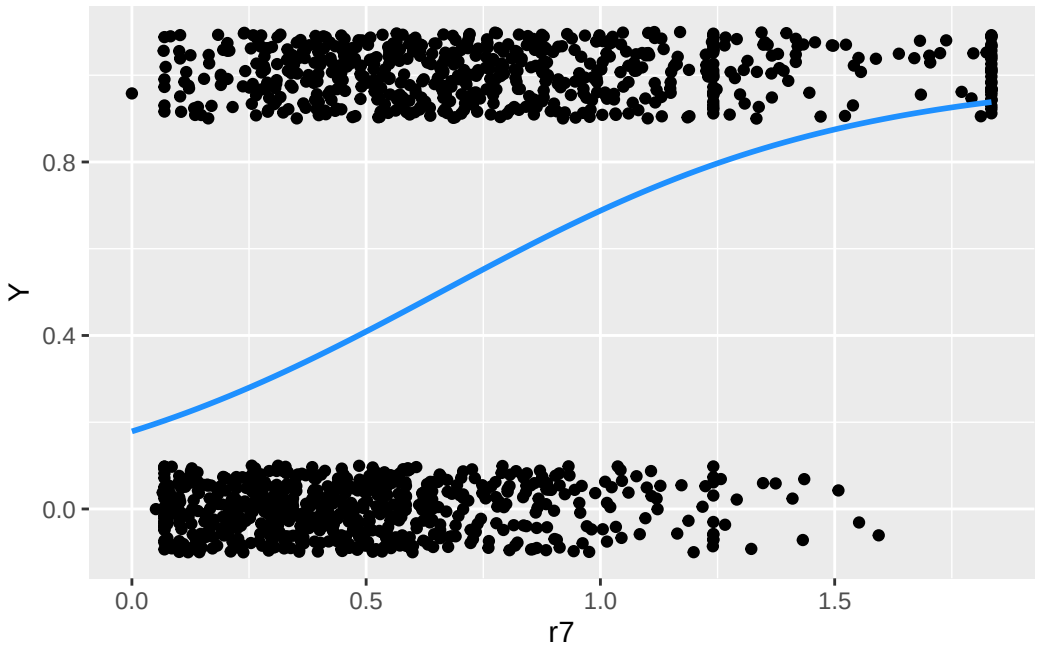

0.8
0.4
0.0
Y
0.00
0.25
0.50
0.75
r2

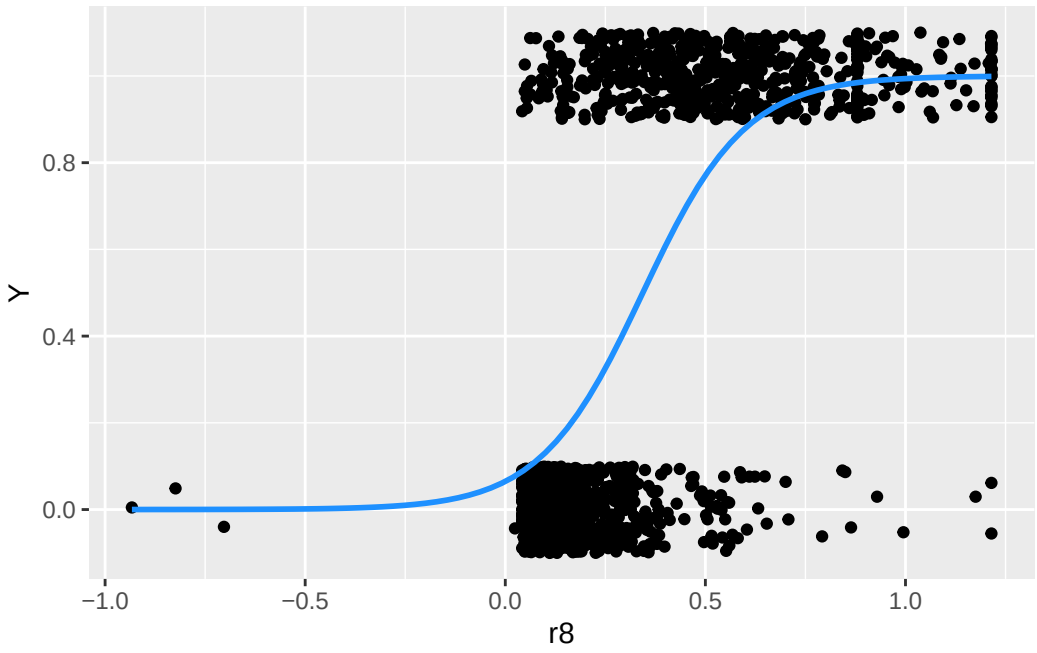

0.8
0.4
0.0
Y
−1.5
−1.0
−0.5
0.0
0.5
r3

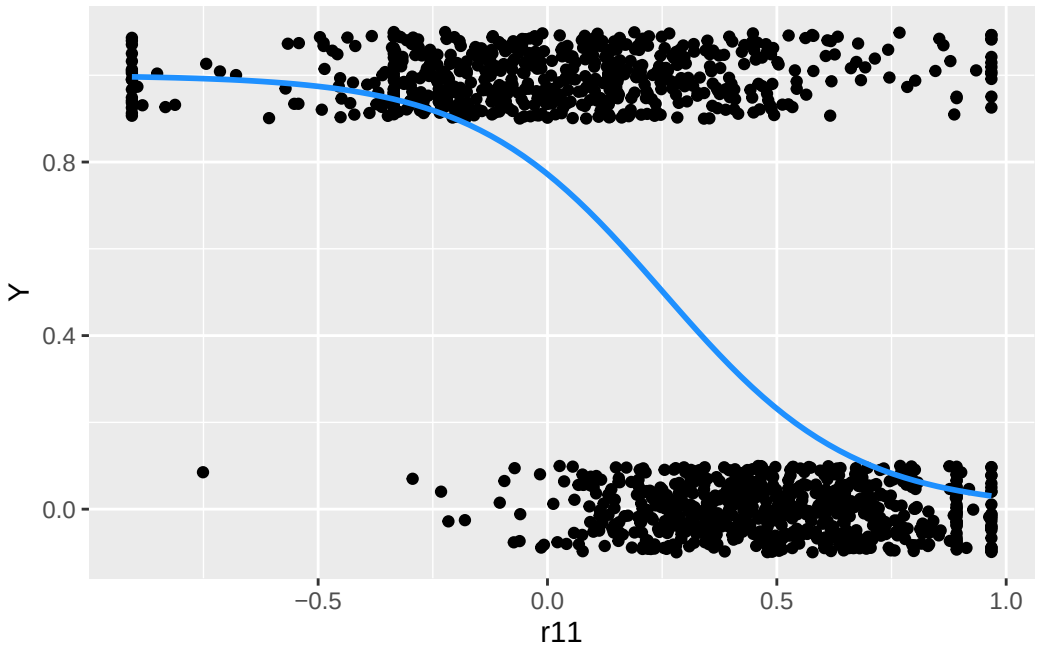

0.8
0.4
0.0
Y
−0.50
−0.25
0.00
0.25
0.50
r4

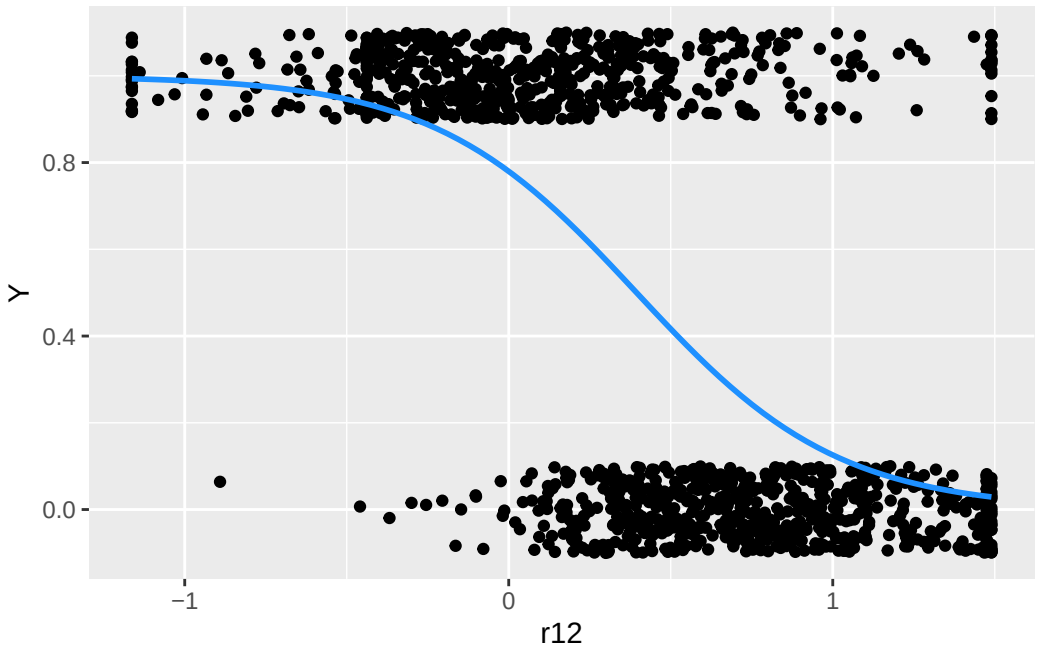

0.8
0.4
0.0
Y
0.0
0.2
0.4
0.6
0.8
r5

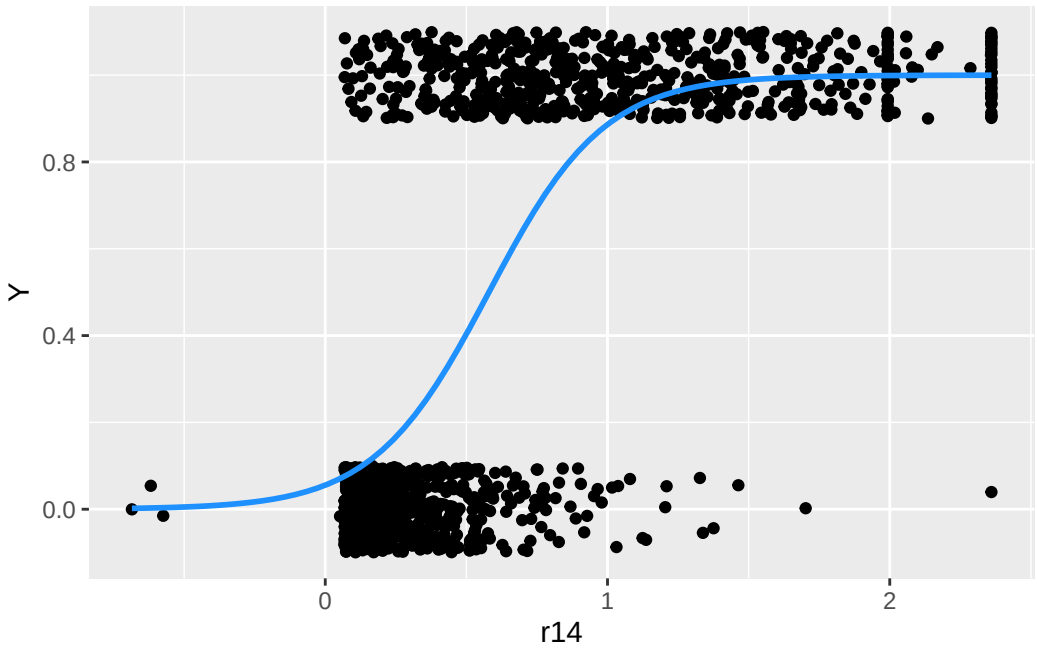

0.8
0.4
0.0
Y
1
2
r6

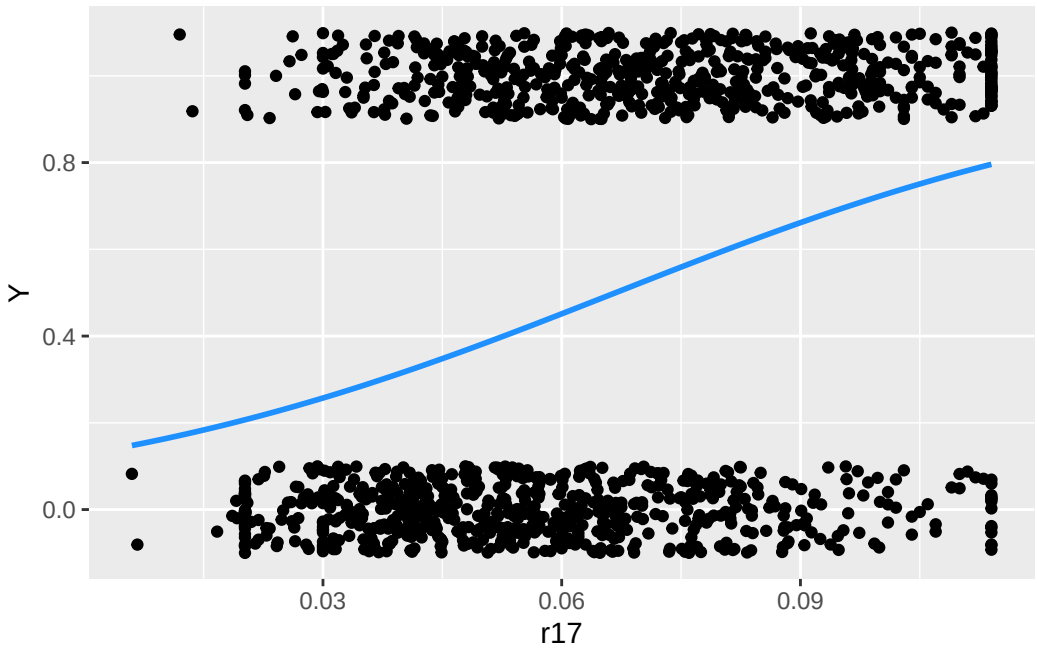

0.8
0.4
0.0
Y
0.0
0.5
1.0
1.5
r7

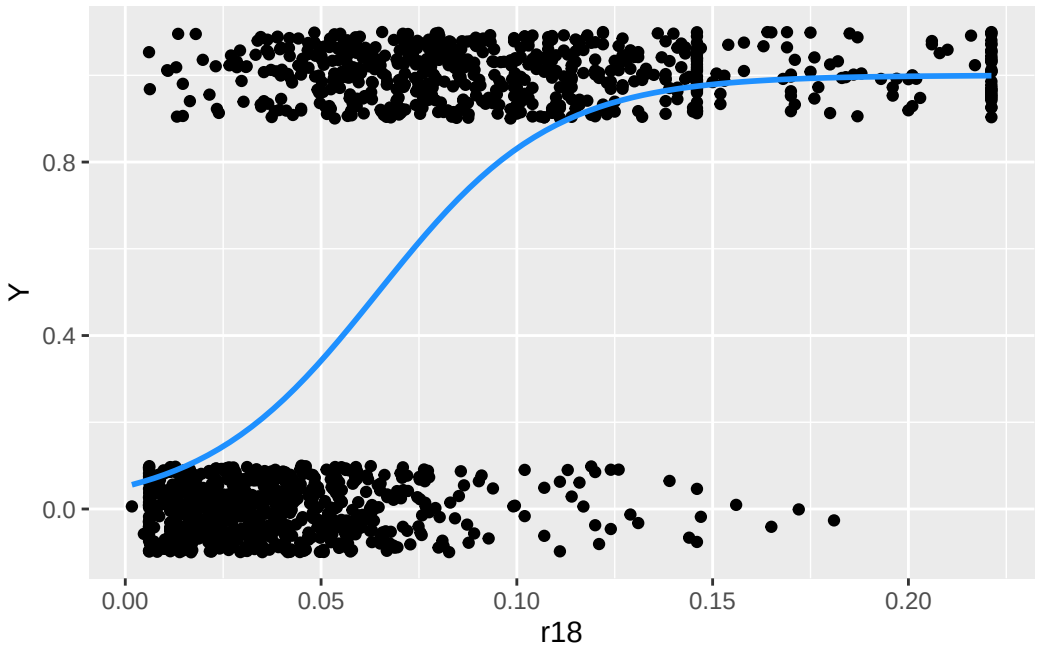

0.8
0.4
0.0
Y
−1.0
−0.5
0.0
0.5
1.0
r8

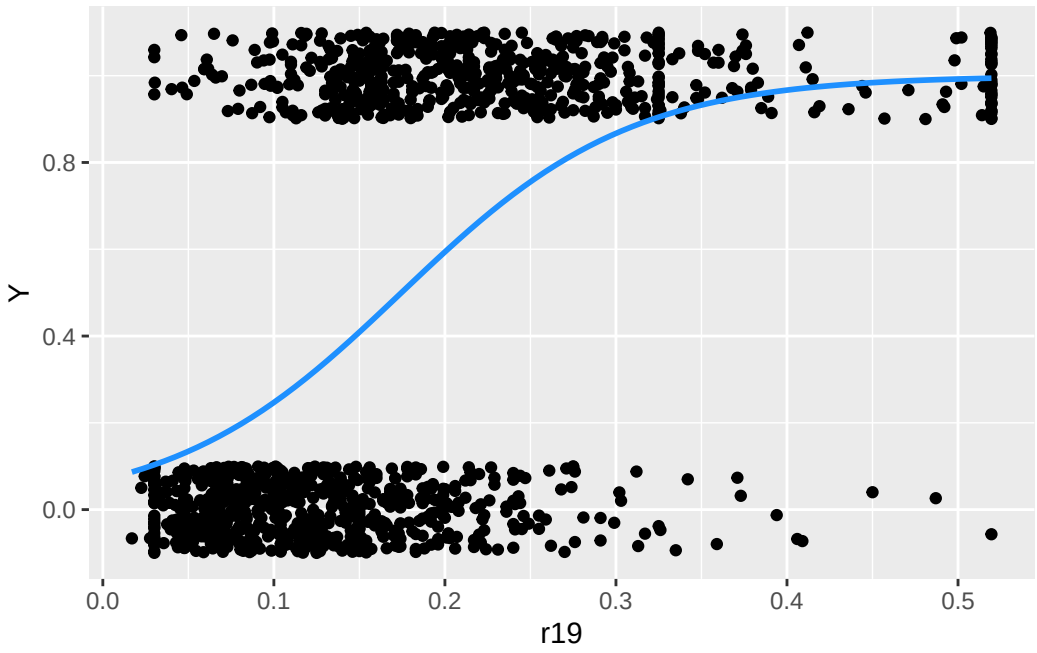

0.8
0.4
0.0
Y
−0.5
0.0
0.5
1.0
r11

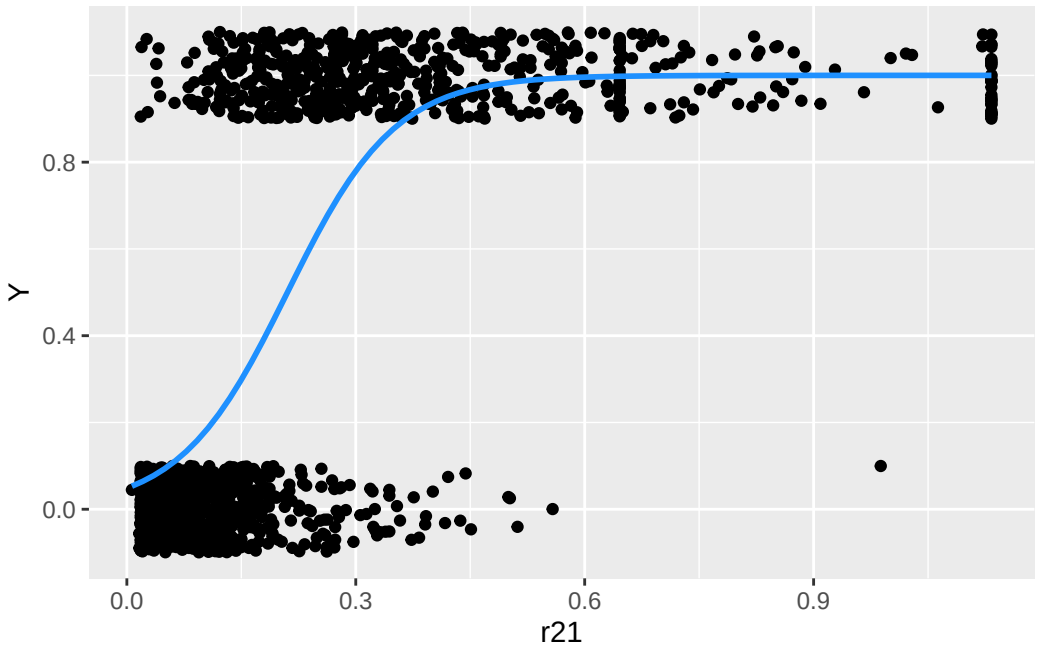

0.8
0.4
0.0
Y
−1
0
1
r12

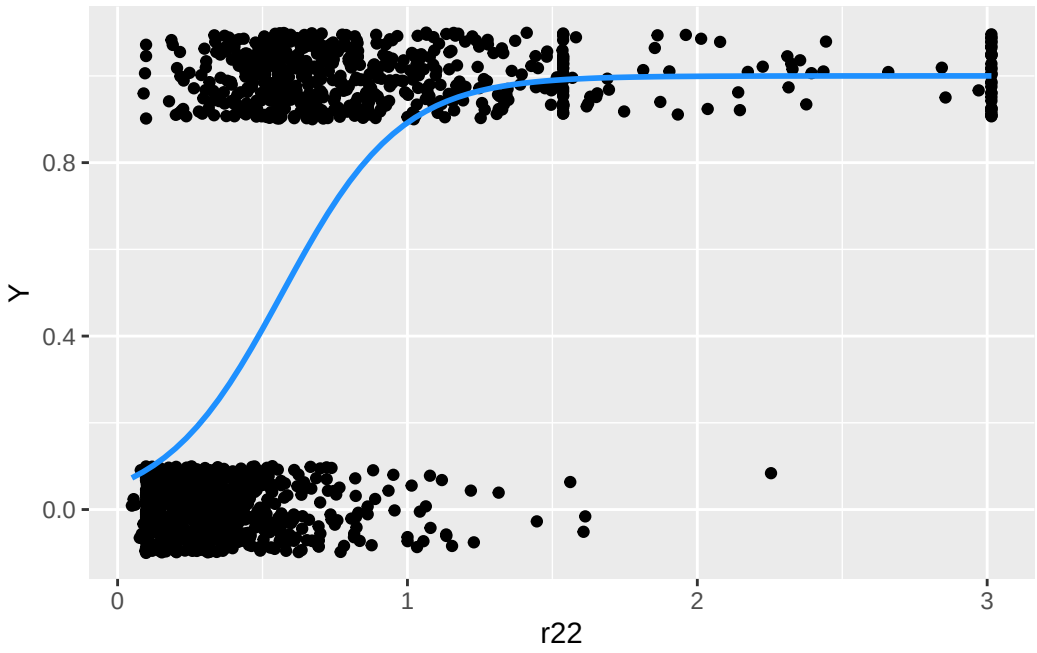

0.8
0.4
0.0
Y
0
1
2
r14

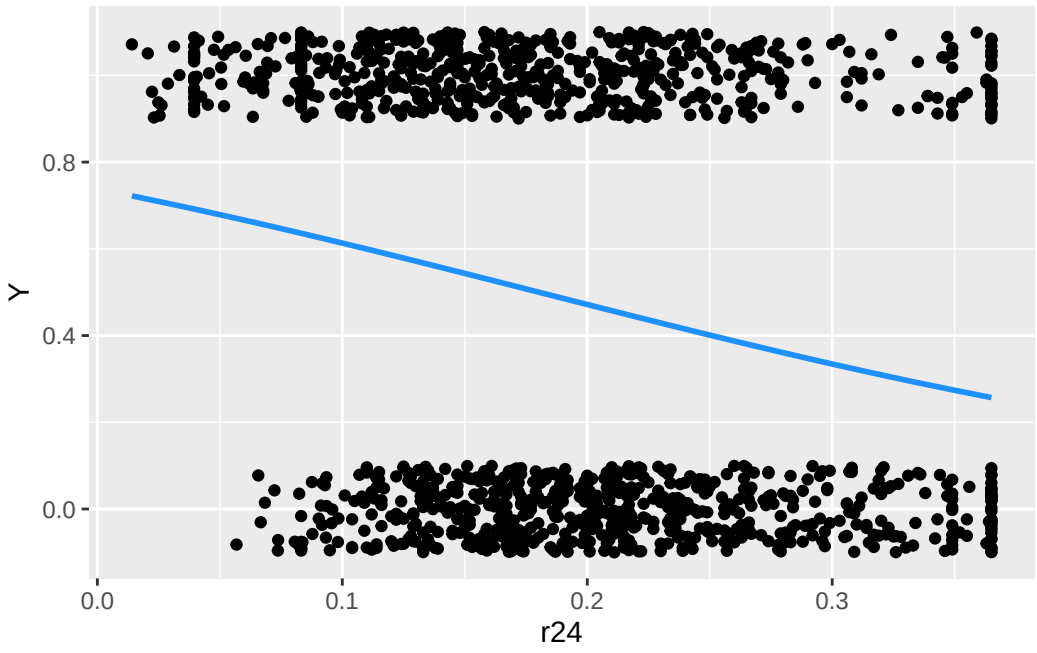

0.8
0.4
0.0
Y
0.03
0.06
0.09
r17

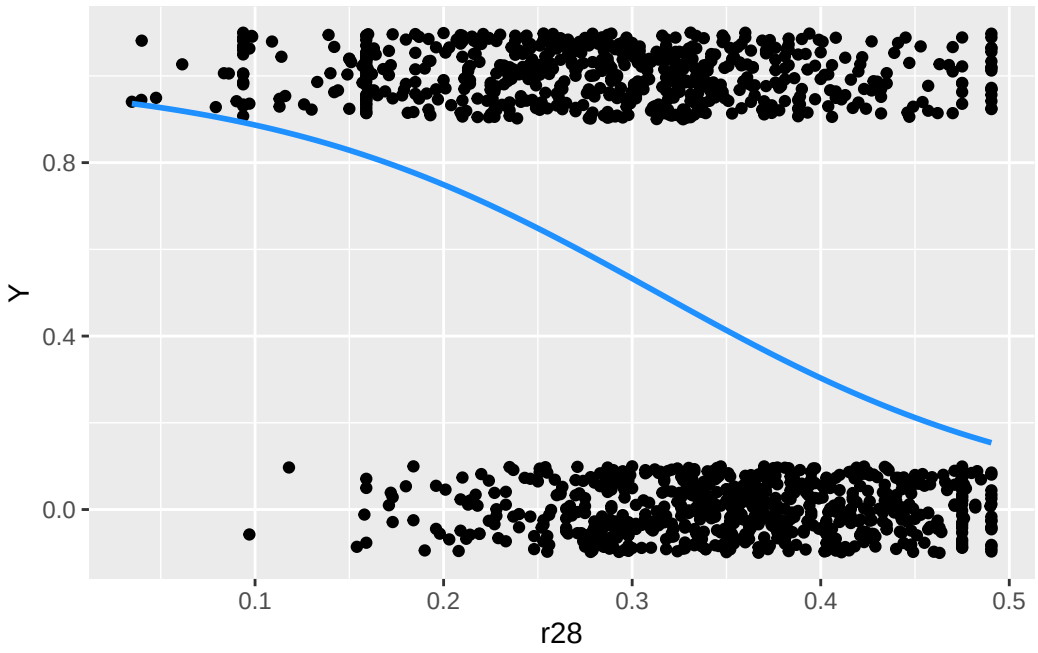

0.8
0.4
0.0
Y
0.00
0.05
0.10
0.15
0.20
r18

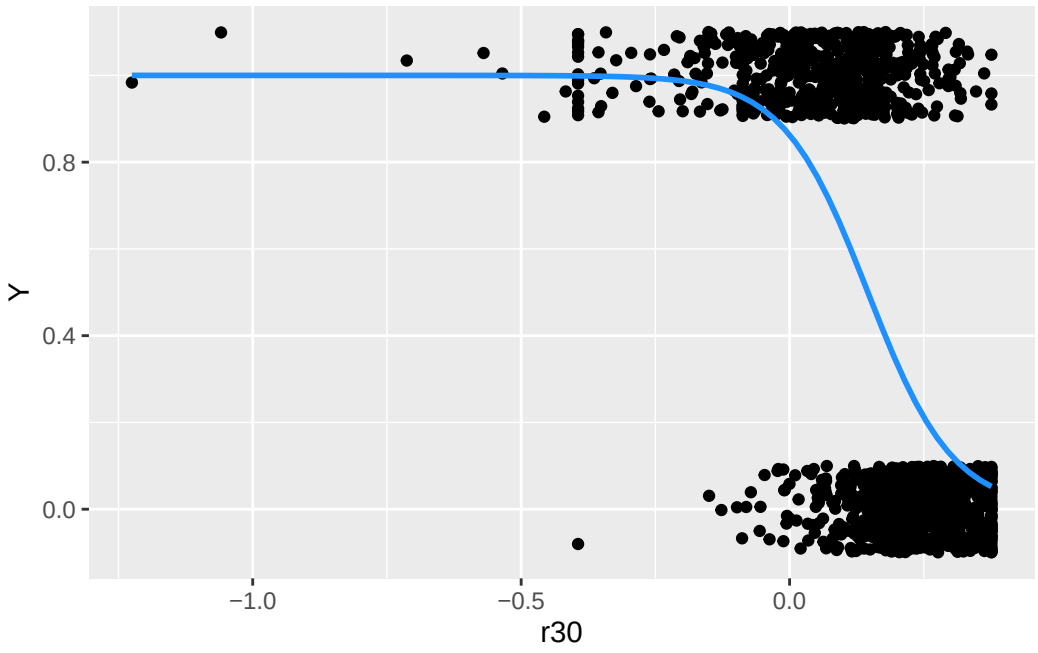

0.8
0.4
0.0
Y
0.0
0.1
0.2
0.3
0.4
0.5
r19

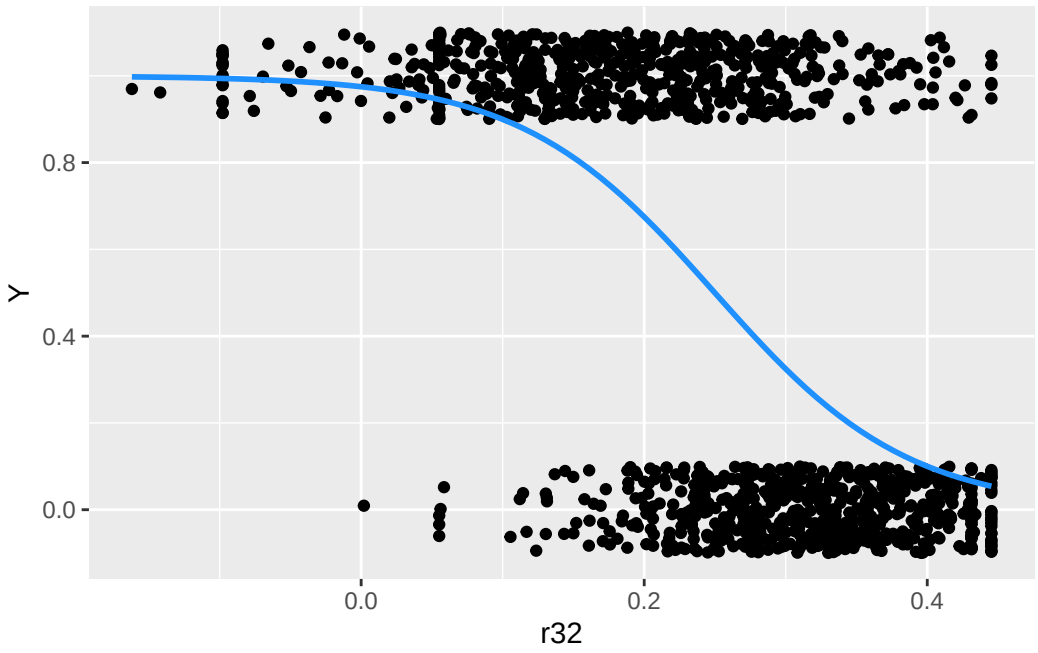

0.8
0.4
0.0
Y
0.0
0.3
0.6
0.9
r21

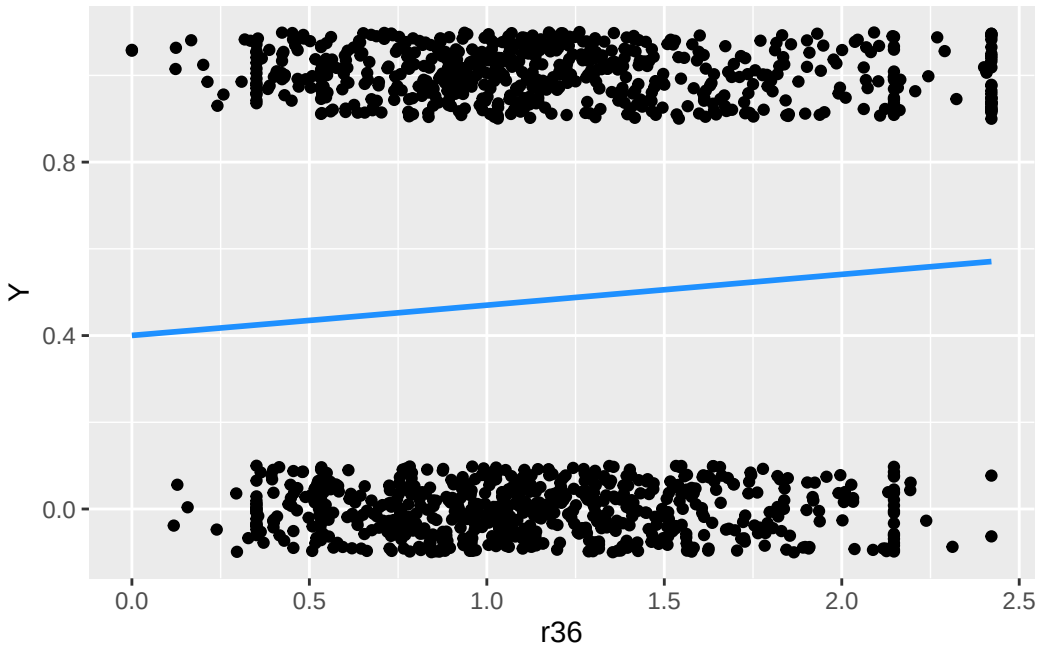

0.8
0.4
0.0
Y
0
1
2
3
r22

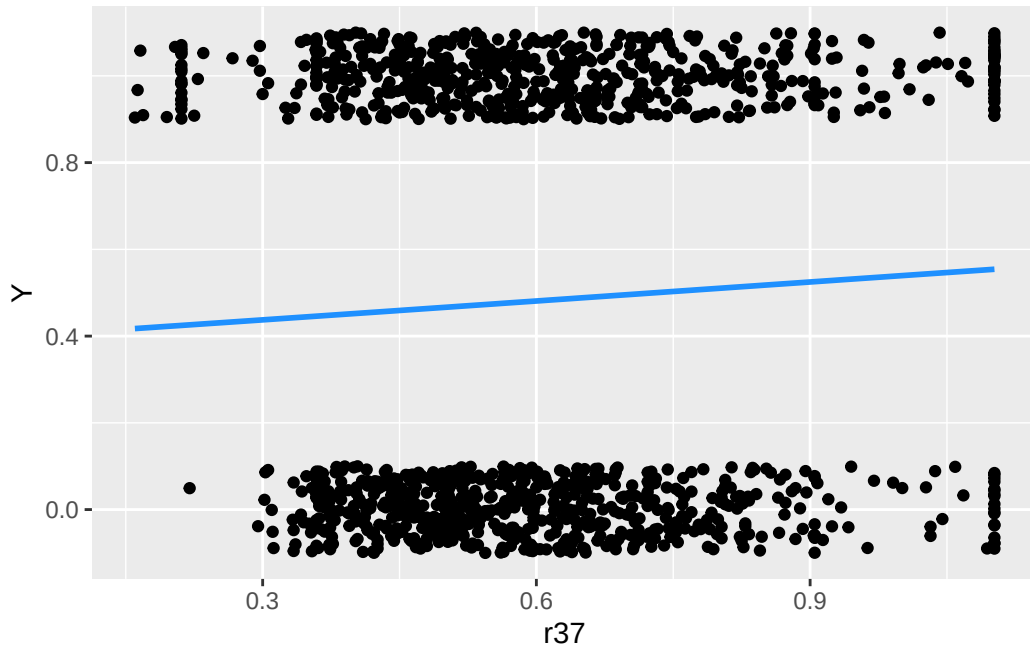

0.8
0.4
0.0
Y
0.0
0.1
0.2
0.3
r24


0.8
0.4
0.0
Y
0.1
0.2
0.3
0.4
0.5
r28


0.8
0.4
0.0
Y
−1.0
−0.5
0.0
r30

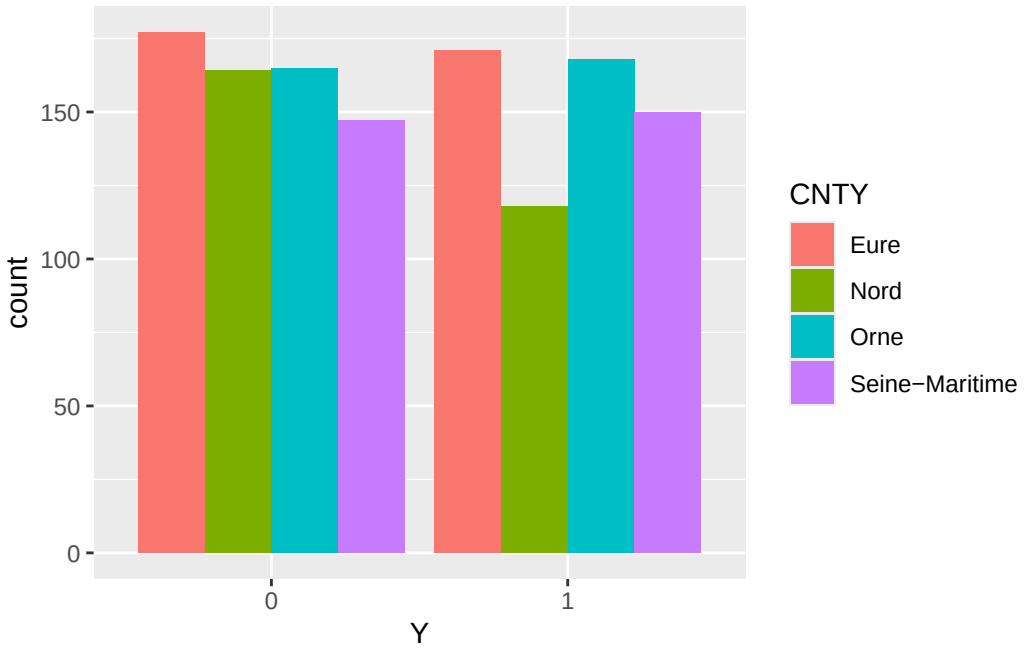

0.8
0.4
0.0
Y
0.0
0.2
0.4
r32

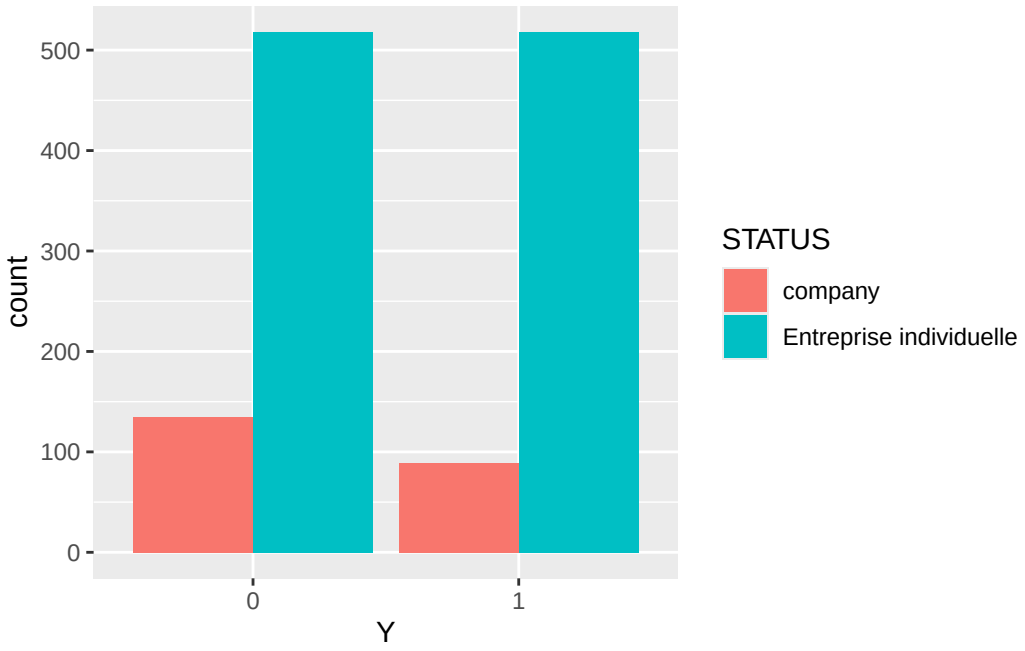

0.8
0.4
0.0
Y
0.0
0.5
1.0
1.5
2.0
2.5
r36

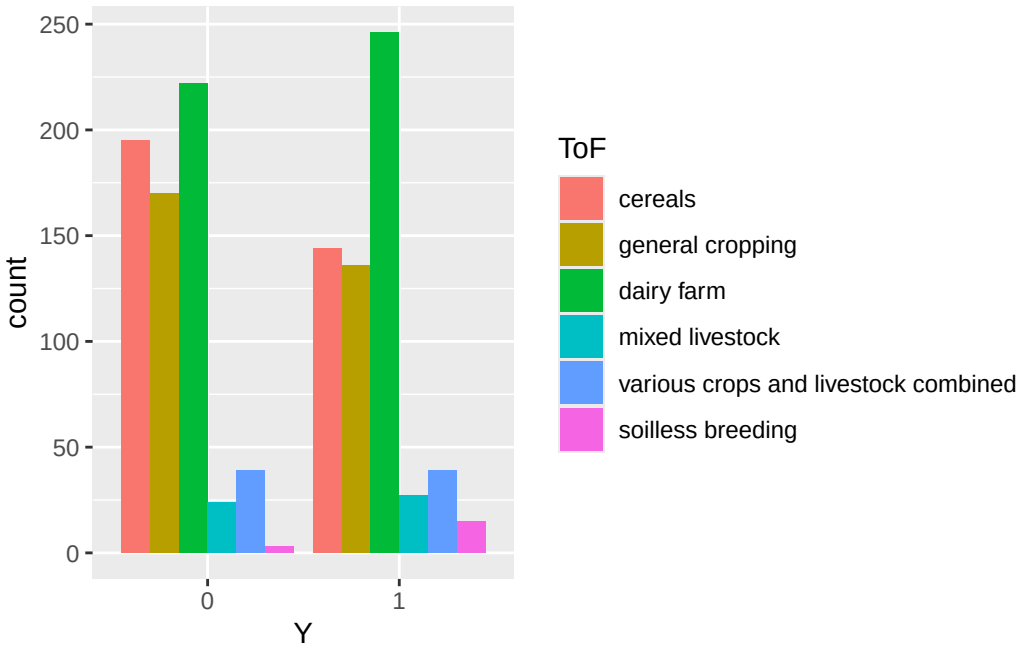


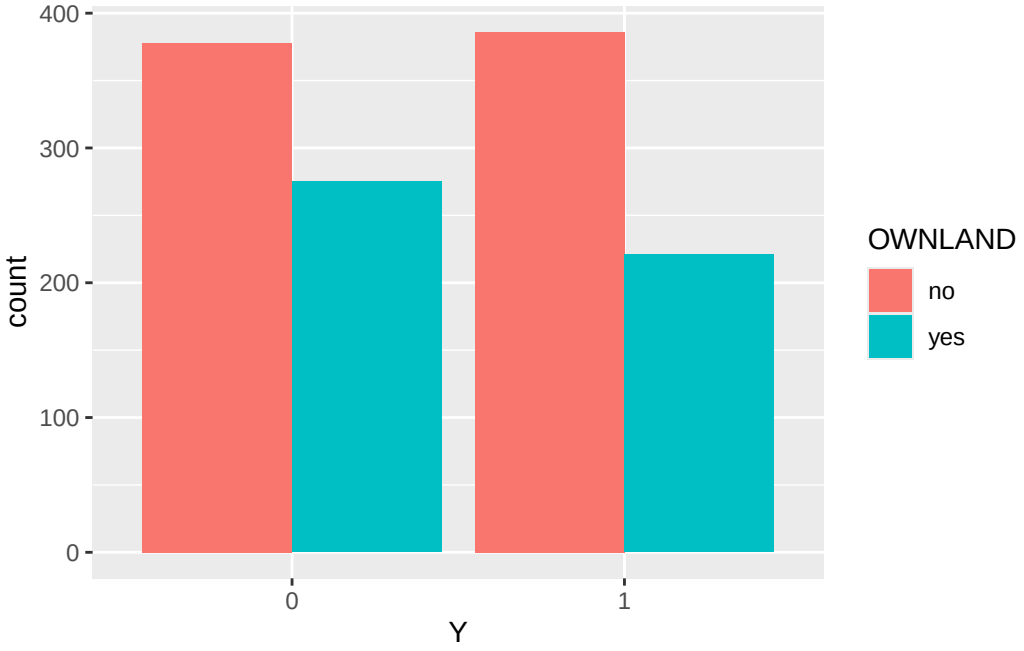
We also show the prevalence of default for each categorical variables:

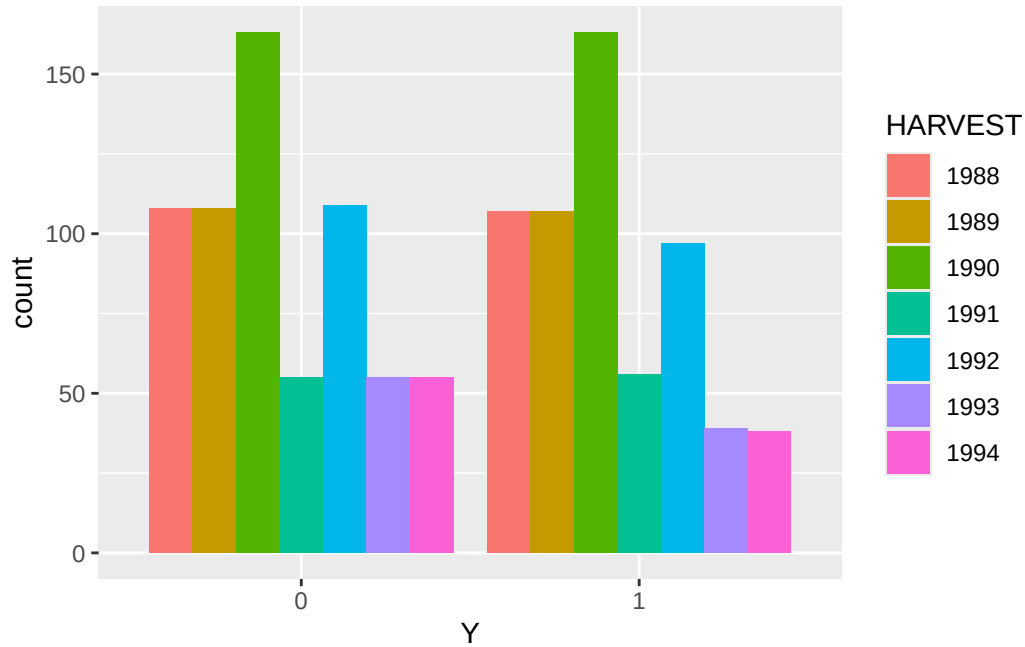
```
vars_quali <- names(don_desbois %>% select_if(is.factor) %>% select(-Y))
for(var in vars_quali){
    var <- as.name(var)
    print(ggplot(don_desbois %>%
                     group_by(!!var, Y) %>%
                     summarize(count = n()) %>%
                     ungroup()) +
        geom_bar(aes(x = Y, y = count, fill = !!var), position="dodge",stat="identity"))
}
```


count
150
100
50
0
0
1
Y
CNTY
Eure
Nord
Orne
Seine−Maritime


count
500
400
300
200
100
0
0
1
Y
STATUS
company
Entreprise individuelle


250
200
150
100
50
0
count
0
1
Y
ToF
cereals
general cropping
dairy farm
mixed livestock
various crops and livestock combined
soilless breeding


400
300
200
100
0
count
0
1
Y
OWNLAND
no
yes



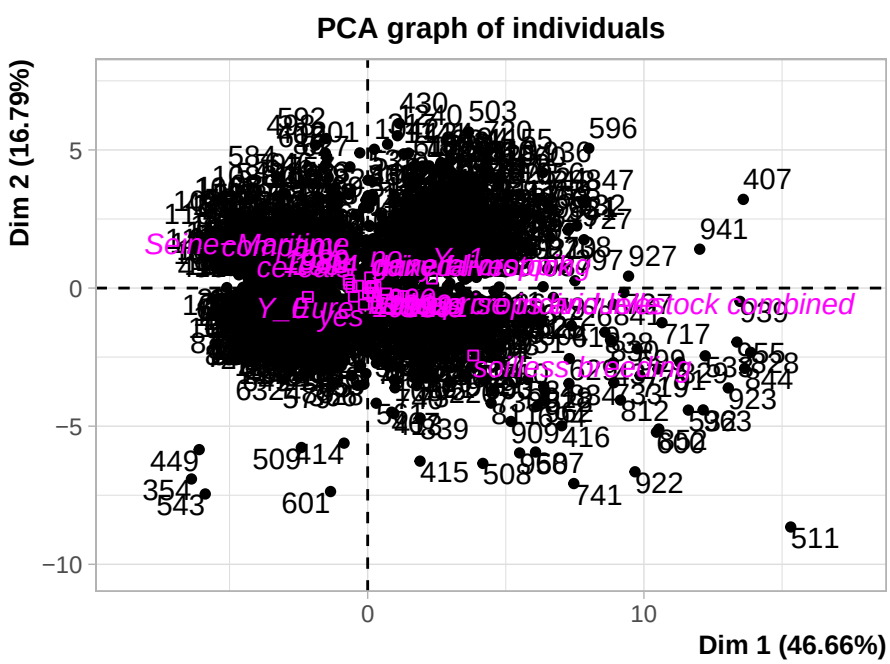

```
class_width <- 0.01
(don_desbois_binned <- don_desbois %>%
    mutate(r17_bins = cut(r17, breaks = seq(0, 0.2, class_width),
                                right = FALSE, dig.lab = 4, include.lowest = TRUE),
           min = floor(r17 / class_width) * class_width,
           max = if_else(r17 == 0 , 1,
                         # customers with 0$ balance should belong to [0, width) class
                         # or be excluded
                         ceiling(r17 / class_width))  * class_width) %>%
    group_by(r17_bins, min, max, Y) %>%
    summarize(n=n()) %>%
    ungroup() %>%
    pivot_wider(names_from = Y, values_from = n) %>%
    replace_na(list(`0` = 0, `1` = 0)) %>%
    mutate(`Mean(Y)` = round(`1` / (`1` + `0`), 4)))
```

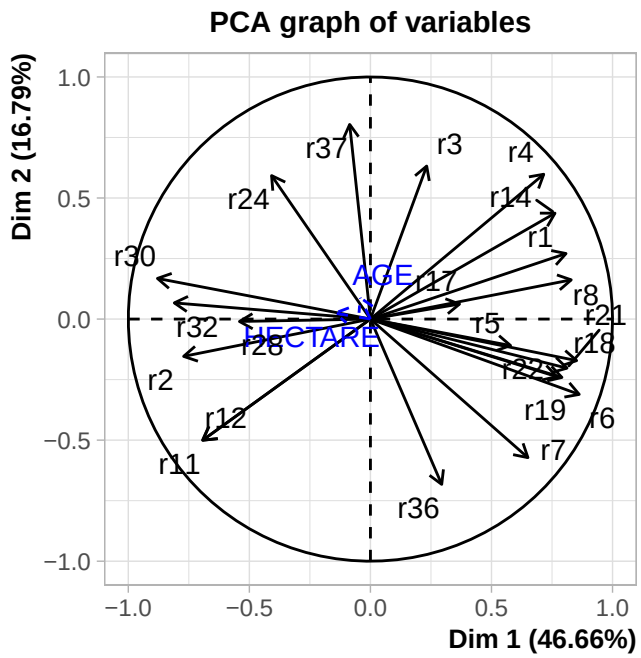
```
# A tibble: 21 x 6
   r17_bins      min   max    `0`    `1` `Mean(Y)`
   <fct>       <dbl> <dbl> <int> <int>     <dbl>
 1 [0,0.01)      0     0.01     2     0     0
 2 [0.01,0.02)   0.01  0.02     4     2     0.333
 3 [0.02,0.03)   0.02  0.03    65    14     0.177
```

```
 4 [0.03,0.04)  0.03  0.03    13     5     0.278
 5 [0.03,0.04)  0.03  0.04    93    33     0.262
 6 [0.04,0.05)  0.04  0.04     1     0     0
 7 [0.04,0.05)  0.04  0.05   111    64     0.366
 8 [0.05,0.06)  0.05  0.05     0     2     1
 9 [0.05,0.06)  0.05  0.06   104    75     0.419
10 [0.06,0.07)  0.06  0.06     2     2     0.5
# i 11 more rows
```

9.2 Performing PCA - A first Scoring function

```
res.pca = FactoMineR::PCA(don_desbois,
                          scale.unit = TRUE,
                          quanti.sup = c(3, 6),
                          quali.sup = c(1, 2, 4, 5, 7, 30),
                          ncp = 5, graph=TRUE)
```






```
FactoMineR::dimdesc(res.pca, axes=c(1,2))
```

```
$Dim.1

Link between the variable and the continuous variables (R-square)
=================================================================================
        correlation       p.value
r6       0.86236020  0.000000e+00
r18      0.85168887  0.000000e+00
r8       0.82938111 2.330014e-320
r21      0.81179591 3.426092e-296
r1       0.80784695 4.105182e-291
r19      0.79096025 1.171331e-270
r22      0.76939141 3.940487e-247
r14      0.76232304 5.666254e-240
r4       0.71581150 1.980339e-198
r7       0.65046752 1.848309e-152
r5       0.57879615 1.385393e-113
r17      0.36806139  1.044336e-41
r36      0.29436936  1.316397e-26
r3       0.23245366  6.387671e-17
r37     -0.08565625  2.341765e-03
```

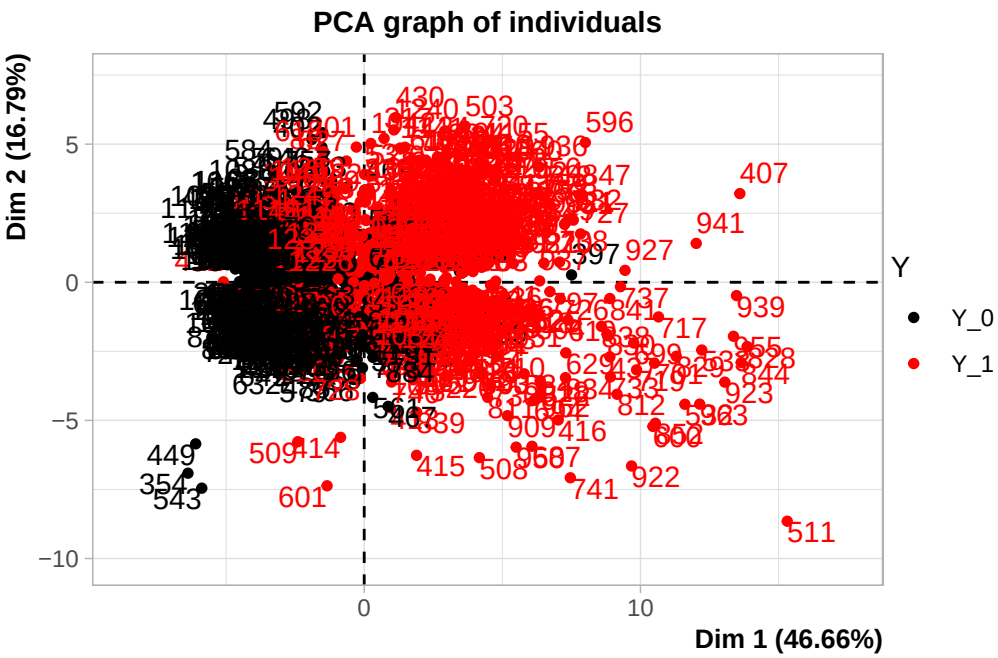
```
HECTARE -0.14744637  1.461126e-07
r24     -0.40817043  9.021914e-52
r28     -0.54086794  1.205880e-96
r12     -0.69202193 2.869670e-180
r11     -0.69331390 3.289759e-181
r2      -0.77151274 2.495616e-249
r32     -0.80993226 8.838285e-294
r30     -0.87962552  0.000000e+00

Link between the variable and the categorical variable (1-way anova)
=============================================
                R2       p.value
Y       0.49111331 9.348303e-187
CNTY    0.04786760  2.632432e-13
HARVEST 0.03797662  9.010371e-09
ToF     0.03103923  1.808729e-07
STATUS  0.02056438  3.192974e-07

Link between variable and the categories of the categorical variables
================================================================
                                 Estimate       p.value
Y=Y_1                           2.2467698 9.348303e-187
CNTY=Orne                       1.1119474  1.605640e-13
ToF=soilless breeding           3.1829576  3.046994e-07
STATUS=Entreprise individuelle  0.6008590  3.192974e-07
HARVEST=1991                    1.2225483  7.752996e-06
HARVEST=1992                    0.5904870  1.156062e-03
HARVEST=1988                   -0.7427342  7.411220e-04
CNTY=Eure                      -0.4896532  5.912444e-04
HARVEST=1989                   -0.7589021  5.490028e-04
ToF=cereals                    -1.2266338  7.772544e-05
CNTY=Seine-Maritime            -0.6460362  4.993511e-05
STATUS=company                 -0.6008590  3.192974e-07
Y=Y_0                          -2.2467698 9.348303e-187

$Dim.2

Link between the variable and the continuous variables (R-square)
=================================================================================
    correlation       p.value
r37  0.80380562 4.896206e-286
r3   0.63191245 2.091324e-141
r4   0.59803723 4.610113e-123
```

```
r24  0.59251517 2.814166e-120
r14  0.43731808  5.320349e-60
r1   0.27068407  1.345585e-22
r30  0.16789176  2.019112e-09
r8   0.16197477  7.379643e-09
AGE  0.08801820  1.764159e-03
r32  0.06671616  1.786147e-02
r17  0.06203095  2.767834e-02
r5  -0.10929143  1.013822e-04
r2  -0.15401447  3.921347e-08
r18 -0.17158238  8.785706e-10
r21 -0.20257647  3.899528e-13
r22 -0.23768202  1.216741e-17
r19 -0.24106163  4.077362e-18
r6  -0.31116423  1.088798e-29
r12 -0.49999563  1.167196e-80
r11 -0.50040837  8.248175e-81
r7  -0.57223645 1.700747e-110
r36 -0.68267475 1.311992e-173

Link between the variable and the categorical variable (1-way anova)
=============================================
                R2      p.value
OWNLAND 0.06253224 2.039288e-19
CNTY    0.06137746 3.760278e-17
STATUS  0.03525656 1.838523e-11
Y       0.02967307 7.528153e-10
ToF     0.02426095 1.009628e-05
HARVEST 0.01059172 3.761371e-02

Link between variable and the categories of the categorical variables
================================================================
                                 Estimate      p.value
OWNLAND=no                      0.4918036 2.039288e-19
CNTY=Seine-Maritime             0.8334341 6.726846e-19
STATUS=company                  0.4718928 1.838523e-11
Y=Y_1                           0.3312512 7.528153e-10
HARVEST=1989                    0.3253062 1.613745e-02
CNTY=Orne                      -0.3004792 1.856507e-03
CNTY=Eure                      -0.3317426 3.612860e-04
ToF=soilless breeding          -2.0384239 4.852557e-08
Y=Y_0                          -0.3312512 7.528153e-10
STATUS=Entreprise individuelle -0.4718928 1.838523e-11
```

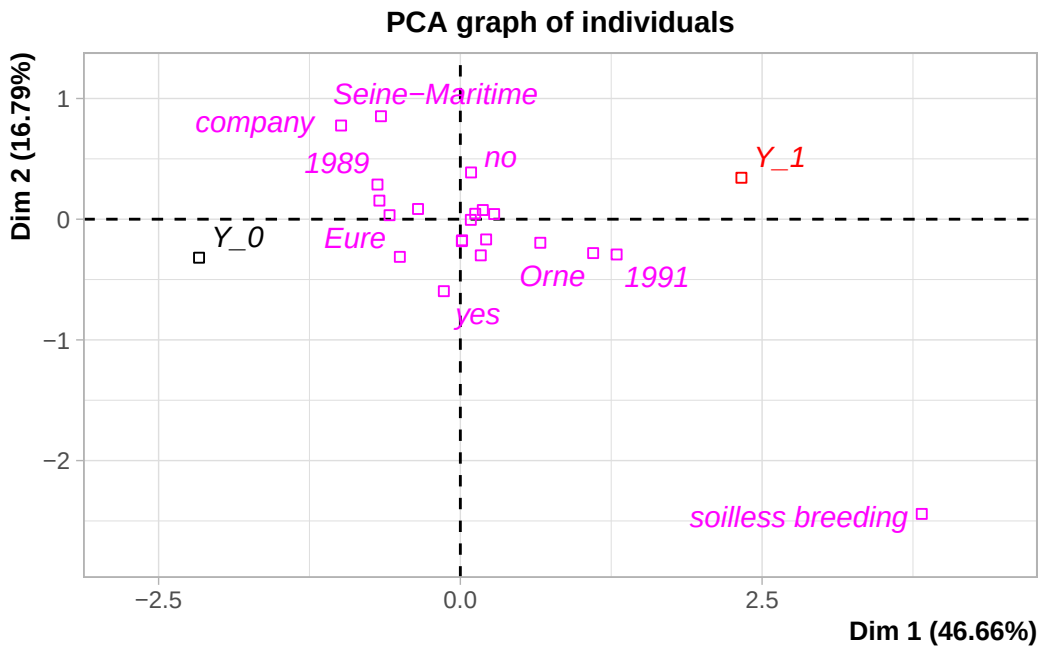
```
OWNLAND=yes                    -0.4918036 2.039288e-19
```

We visualize the farm holdings as a bivariate plot on the first two components of PCA (based on financial ratios), we use illustrative variable Y (0="healthy"; 1="failing"):

```
# Similar to Desbois Fig 2.
# Plot of the farm holdings in the first factorial plane of the normalized PCA based on fi
# with illustrative variable Y (0="healthy"; 1="failing")
FactoMineR::plot.PCA(res.pca, axes=c(1, 2), choix="ind", habillage=30, invisible = c("qual
```




```
FactoMineR::plot.PCA(res.pca, axes=c(1, 2), choix="ind", habillage=30, invisible = c("ind"
```

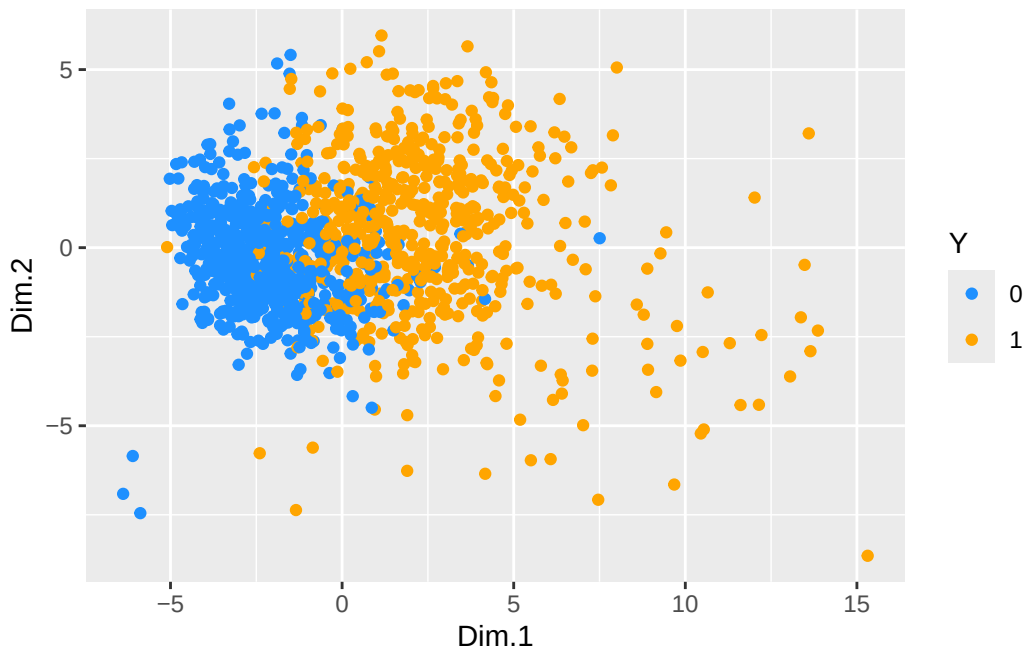


(Extra) As an additional exercise after having fit a classifier in the next questions, it would be nice to visualize its decision boundary on the Principal Component (PC_1/PC_2) graph (using the transformation matrix).

```
# https://stats.stackexchange.com/questions/460787/pcr-after-pca-with-mixed-data-how-to-ex
# https://stats.stackexchange.com/questions/494866/how-to-explain-the-numerical-discrepanc

# https://stats.stackexchange.com/questions/134282/relationship-between-svd-and-pca-how-to
# Extracting first five Principal components from FactomineR object
# (X nxp matrix of Desbois observations, SVD decomposition X = U %*% diag(vs) %*% t(V),
# where U: unitary matrix of left-singular vectors, vs: singular values, V: unitary matrix
# Principal components X %*% V = U %*% diag(vs) %*% t(V) %*% V = U %*% diag(vs)
principal_components_1 <- res.pca$svd$U %*% diag(res.pca$svd$vs[1:5])
# Can also be directly extracted from:
principal_components_2 <- res.pca$ind$coord
# Or using PC = X %*% V (where X has been centered and scaled, base R uses a different sca
X_for_PCA <- don_desbois %>% select(r1:r37) %>% as.matrix()
principal_components_3 <- sqrt(nrow(X_for_PCA) / (nrow(X_for_PCA) - 1)) * scale(X_for_PCA)
```

```
naive_score <-  bind_cols(don_desbois, as_tibble(principal_components_2))
ggplot(naive_score) +
```

```
    geom_point(aes(x = Dim.1, y = Dim.2, col = Y)) +
    scale_colour_manual(values = c("dodgerblue", "orange"))
```



This plot which is comparable with:

Looking at this plot Desbois concludes that a simple classifier can be devised using the first PCA coordinate a classifier (setting a threshold at $PC_1 > 0.02$):

Going further than Desbois and for illustrative purposes only we can use PC_1 as a very simple and rough score function.

```
library("ROCR")

# naive score
pred <- prediction(naive_score$Dim.1, naive_score$Y)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf, main="ROC curve", xlab="Specificity",
     ylab="Sensitivity", col = "darkorange",
     print.cutoffs.at = c(-2.5,-1,0.00,1,5),
     cutoff.label.function = function(x) {paste0("              s = ",round(x,2))})
abline(0, 1) #add a 45 degree line
```

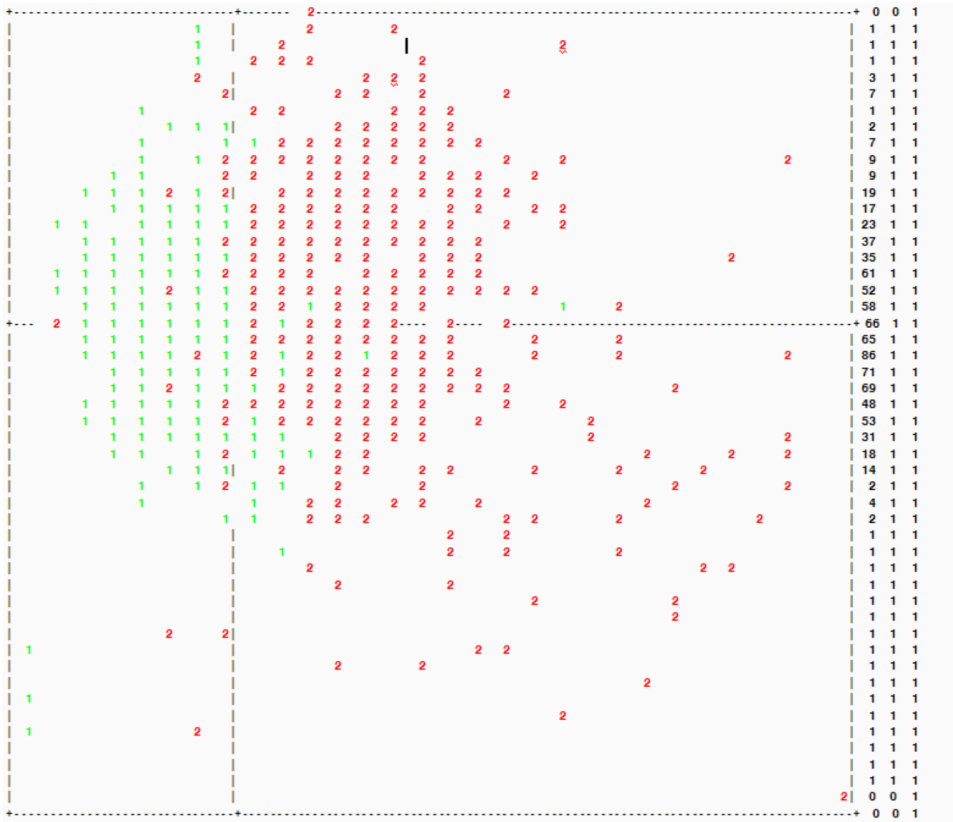


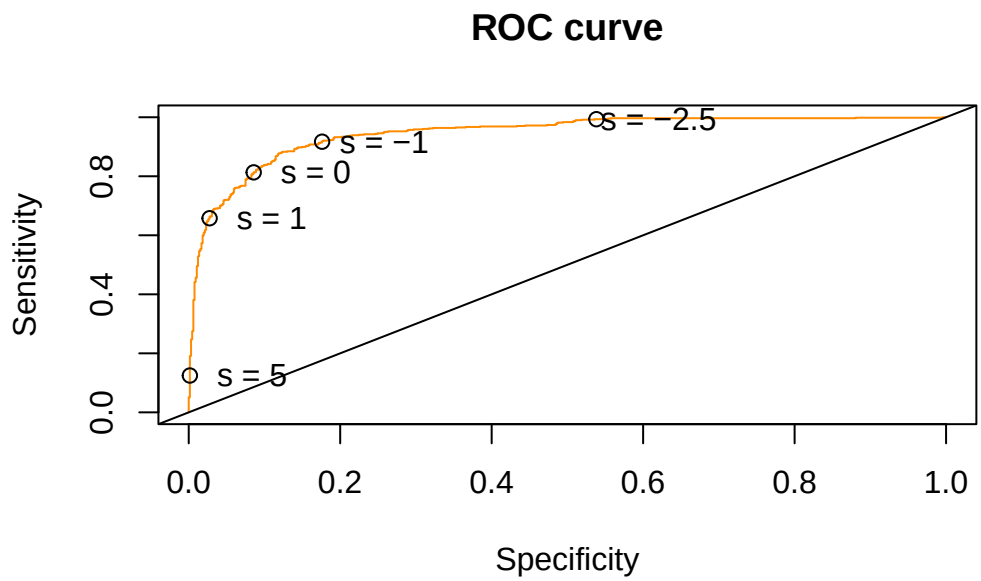
Figure 2. Plot of the farm holdings in the first factorial plane of the normalized PCA based on financial ratios with illustrative variable LDIFF (1="healthy"; 2="failing").

inertia. Considering the respective positions of the healthy and failing groups of farm holdings on the first factorial plane, we see that the *F1* axis could be used as an index of classification (see Figures 2 and 4).

Indeed, the average coordinate of the "healthy" group on the *F1* axis is approximately equal to $\mu_0 \approx -0.6754$ while the average coordinate of the "failing" group is approximately equal to $\mu_1 \approx 0.7266$.

A geometrical rule of classification for a farm holding $\boldsymbol{i}_0$ which does not belong to the training sample consists in positioning the farm holding in relation to the "pivot point", i.e. the median point of the segment linking the two group barycenters on the *F1* axis, that is to say

$$\frac{\mu_0 + \mu_1}{2} = 0.0256:$$



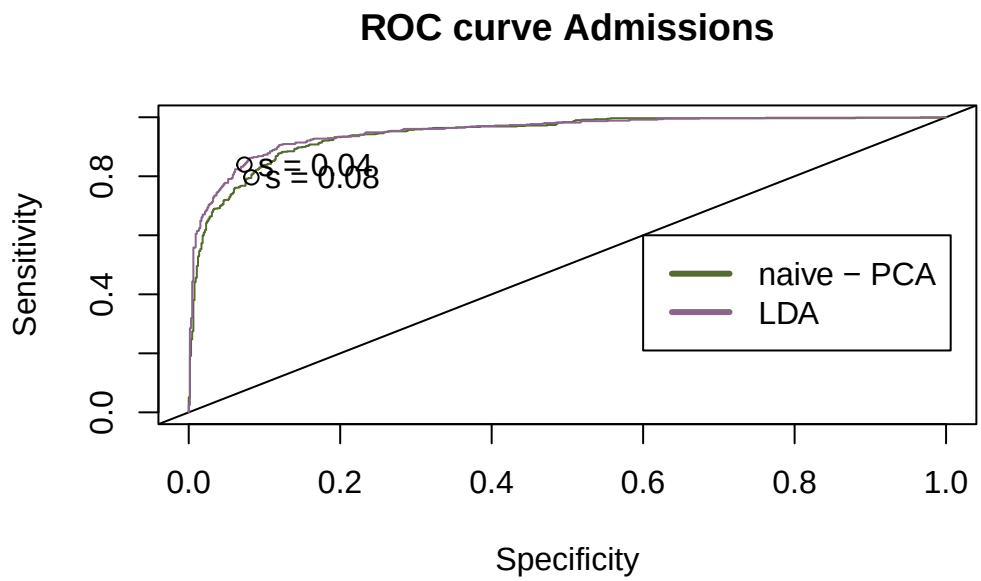
We compare with LDA using Desbois selected variables by a forward procedure:

```
lda_desbois <- MASS::lda(Y~r2+r3+r7+r14+r17+r18+r21+r32+r36, data=don_desbois)
lda_desbois$scaling
```

```
           LD1
r2  -2.5418242
r3   2.3979966
r7   1.0993874
r14  0.6040669
r17 18.7020927
r18 -6.7976937
r21 -0.6221733
r32 -4.1627070
r36  0.1918914
```

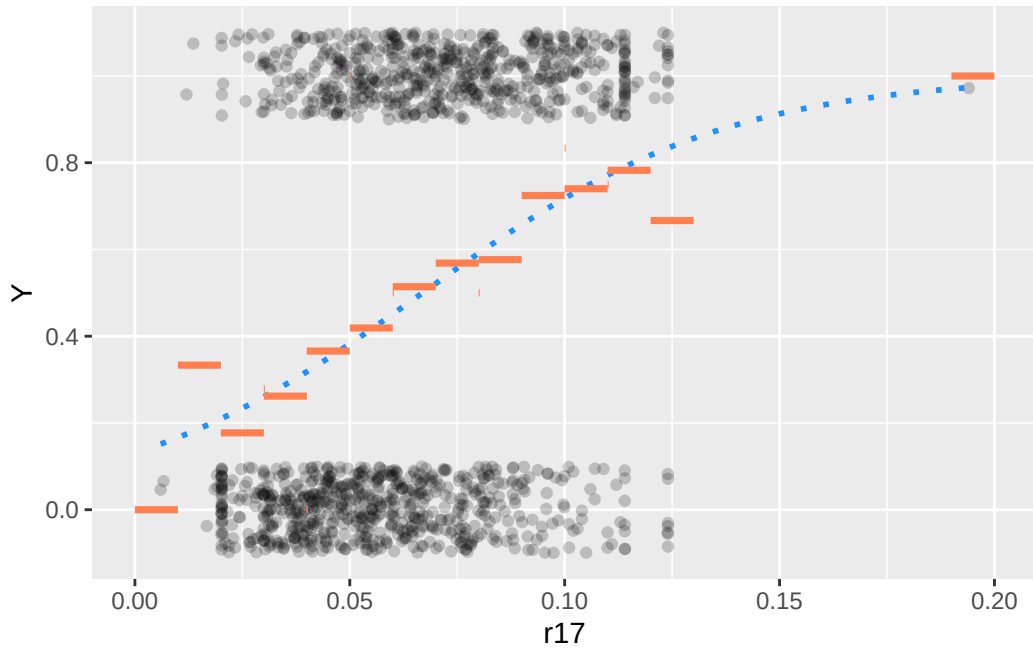
```
# for ROCR
labels_desbois <- don_desbois$Y

# naive score
predictions_naive_score <- naive_score$Dim.1
```

```r
naive_score_group <- naive_score %>%
  group_by(Y) %>%
  summarize(mean_score = mean(Dim.1), n = n())

cutoff_naive <- mean(naive_score_group$mean_score)

pred <- prediction(predictions_naive_score, labels_desbois)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "darkolivegreen",
     print.cutoffs.at = c(cutoff_naive),
     cutoff.label.function = function(x) {paste0("                s = ",round(x,2))})
abline(0, 1) #add a 45 degree line

# lda
predictions_lda <- predict(lda_desbois)$x

lda_score_group <- bind_cols(don_desbois, predictions_lda) %>%
  group_by(Y) %>%
  summarize(mean_score = mean(LD1), n = n())

cutoff_lda <- mean(lda_score_group$mean_score)

pred <- prediction(predictions_lda, labels_desbois)
perf <- performance(pred, measure = "tpr", x.measure = "fpr")
plot(perf, add = TRUE, main="ROC curve Admissions", xlab="Specificity",
     ylab="Sensitivity", col = "plum4",
     print.cutoffs.at = c(cutoff_lda),
     cutoff.label.function = function(x) {paste0("                s = ",round(x,2))})
legend(0.6,0.6,
       c('naive - PCA', 'LDA'),
       col=c("darkolivegreen", "plum4"),lwd=3)
```



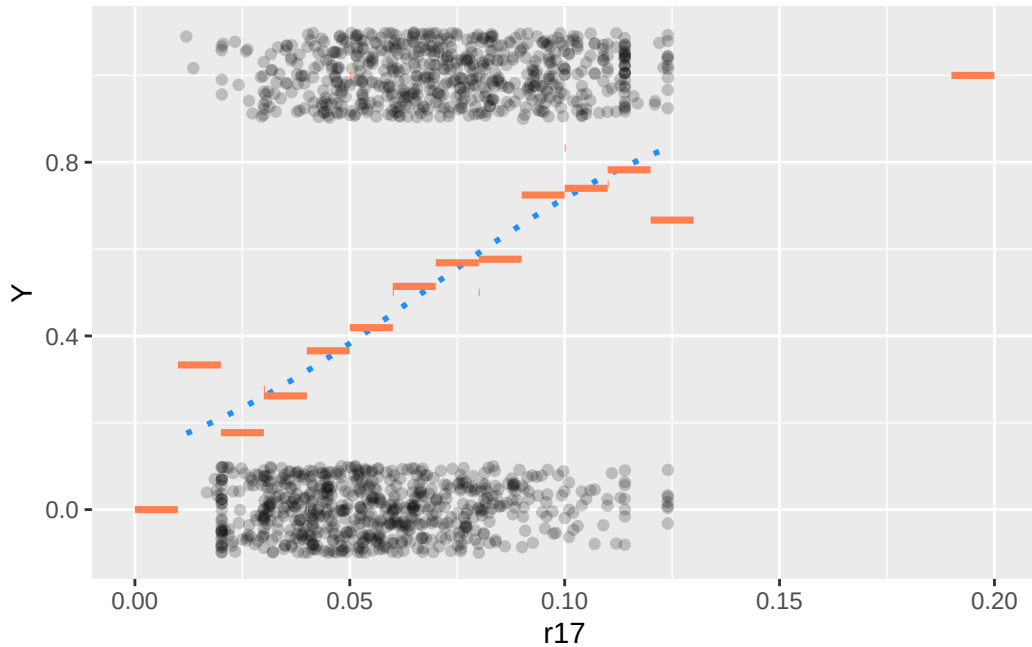
As an illustration we use the discretized ratio r17 and plot the conditional distribution of default given r17 classes (by 0.2) with a univariate logistic regression fitted:

```
(default_occurrence <-
 ggplot(don_desbois %>% mutate(Y = if_else(Y == "1", 1, 0)), aes(x=r17, y=Y)) +
 geom_jitter(alpha=0.2, height=0.1) +
 geom_smooth(method = "glm",
             formula = y ~ x,
             method.args = list(family = "binomial"),
             se = FALSE,
             col = "dodgerblue",
             linetype = "dotted") +
 geom_segment(data = don_desbois_binned,
              aes(x = min, xend = max, y = `Mean(Y)`, yend = `Mean(Y)`),
              color = 'coral',
              linewidth=1.25))
```



Seems to handle quite well some outliers, below removing them for comparison:

```
(default_occurrence <-
 ggplot(don_desbois %>%
          mutate(Y = if_else(Y == "1", 1, 0)) %>%
          filter(r17 < 0.15, r17 > 0.01), aes(x=r17, y=Y)) +
 geom_jitter(alpha=0.2, height=0.1) +
 geom_smooth(method = "glm",
             formula = y ~ x,
             method.args = list(family = "binomial"),
             se = FALSE,
             col = "dodgerblue",
             linetype = "dotted") +
 geom_segment(data = don_desbois_binned,
              aes(x = min, xend = max, y = `Mean(Y)`, yend = `Mean(Y)`),
              color = 'coral',
              linewidth=1.25))
```



9.3 Coefficient interpretation

Fitting a Logistic Regression model with all variables in data set, interpret the `r17` coefficient:

```
# YOUR CODE HERE
```

```
glm_desbois_full <- glm(Y~., data = don_desbois, family=binomial())
summary(glm_desbois_full)
```

```
Call:
glm(formula = Y ~ ., family = binomial(), data = don_desbois)

Coefficients:
                                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                              -6.163042   5.650545  -1.091 0.275405
CNTYNord                                 -2.202261   0.965280  -2.281 0.022520
CNTYOrne                                 -2.088896   0.650774  -3.210 0.001328
CNTYSeine-Maritime                       -0.282046   0.406908  -0.693 0.488219
STATUSEntreprise individuelle            -1.368038   0.399409  -3.425 0.000614
HECTARE                                  -0.037567   0.004741  -7.924  2.3e-15
```

ToFgeneral cropping	-0.950681	0.405734	-2.343	0.019124
ToFdairy farm	-0.093208	0.339259	-0.275	0.783515
ToFmixed livestock	-1.339574	0.655521	-2.044	0.041000
ToFvarious crops and livestock combined	-0.160108	0.810691	-0.197	0.843439
ToFsoilless breeding	0.542009	1.357896	0.399	0.689780
OWNLANDyes	-0.838653	0.321468	-2.609	0.009085
AGE	0.050642	0.017302	2.927	0.003422
HARVEST1989	0.003303	0.388351	0.009	0.993214
HARVEST1990	-0.260576	0.390947	-0.667	0.505075
HARVEST1991	-0.431184	0.818164	-0.527	0.598184
HARVEST1992	-0.958839	0.864309	-1.109	0.267271
HARVEST1993	-1.062076	1.022213	-1.039	0.298806
HARVEST1994	-0.375565	1.036817	-0.362	0.717181
r1	26.854618	11.806312	2.275	0.022930
r2	-0.271011	4.681730	-0.058	0.953839
r3	3.612654	2.098883	1.721	0.085210
r4	-6.141536	11.138427	-0.551	0.581371
r5	-16.888857	12.195982	-1.385	0.166117
r6	-3.842933	2.045387	-1.879	0.060268
r7	5.779176	2.456109	2.353	0.018624
r8	-0.169082	2.667016	-0.063	0.949450
r11	-3.037724	2.762245	-1.100	0.271450
r12	3.001803	1.896926	1.582	0.113545
r14	1.793328	0.875458	2.048	0.040516
r17	45.338851	11.957427	3.792	0.000150
r18	-27.530417	21.881477	-1.258	0.208334
r19	6.934941	13.690645	0.507	0.612473
r21	0.831926	4.903891	0.170	0.865288
r22	1.006209	2.164071	0.465	0.641959
r24	3.060396	6.935258	0.441	0.659010
r28	-4.709086	13.910513	-0.339	0.734966
r30	9.676557	14.306553	0.676	0.498804
r32	-13.483358	17.560030	-0.768	0.442580
r36	1.753279	0.548686	3.195	0.001396
r37	-3.919677	2.225891	-1.761	0.078247

(Intercept)	
CNTYNord	*
CNTYOrne	**
CNTYSeine-Maritime	
STATUSEntreprise individuelle	***
HECTARE	***
ToFgeneral cropping	*

```
ToFdairy farm
ToFmixed livestock                      *
ToFvarious crops and livestock combined
ToFsoilless breeding
OWNLANDyes                              **
AGE                                     **
HARVEST1989
HARVEST1990
HARVEST1991
HARVEST1992
HARVEST1993
HARVEST1994
r1                                      *
r2
r3                                      .
r4
r5
r6                                      .
r7                                      *
r8
r11
r12
r14                                     *
r17                                     ***
r18
r19
r21
r22
r24
r28
r30
r32
r36                                     **
r37                                     .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1745.05  on 1259  degrees of freedom
Residual deviance:  486.24  on 1219  degrees of freedom
AIC: 568.24
```

```
Number of Fisher Scoring iterations: 8
```

The coefficient of `r17` is 45.34. Hence an increase of `r17` by 1% increases odds for default by a factor of 1.57.

9.4 Stepwise Logistic Regression

Use forward stepwise selection based on the AIC criterion to select variables in the Agriculture Farm Lending data set. Compare with results from the article. (Extra, perform the same analysis using LDA/Wilks Lambda as in the article).

```
# YOUR CODE HERE
```

```
#
# Due to outliers some predictions "saturate"  to 0 or 1
# Warning: glm.fit: fitted probabilities numerically 0 or 1 occurredWarning: glm.fit: fitt

# winsorizing financial ratios removes the issue
don_desbois_winsorized <- don_desbois %>%
    mutate(across(r1:r37, ~ DescTools::Winsorize(.x , quantile(.x, probs = c(0.025, 0.975)
# data_afl <- don_desbois_winsorized
```

We first define lower (intercept only) / upper (all variables) bound models for the stepwise approaches:

```
data_afl <- don_desbois

#define intercept-only model
intercept_only <- glm(Y ~ 1, data=data_afl, family="binomial")

#define model with all predictors
# In Desbois only financial ratios are used
all <- glm(Y ~ ., data=data_afl %>% select(Y, starts_with('r')), family="binomial")

# We use all variables
# all <- glm(Y ~ ., data=data_afl , family="binomial")
```

Then we use the `stats::step()` function to perform a forward stepwise selection based on AIC:

```r
# perform forward stepwise regression
forward_aic <- step(intercept_only, direction='forward', scope=formula(all), k=2, trace =
summary(forward_aic)
```

```
Call:
glm(formula = Y ~ r1 + r21 + r14 + r17 + r24 + r11 + r28 + r36 +
    r3 + r8 + r32, family = "binomial", data = data_afl)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -11.1573     1.3883  -8.036 9.25e-16 ***
r1           10.3995     1.1853   8.773  < 2e-16 ***
r21           3.9664     1.5931   2.490 0.012787 *
r14           2.4414     0.6955   3.510 0.000448 ***
r17          36.0464     7.2134   4.997 5.82e-07 ***
r24          -4.1232     2.8366  -1.454 0.146058
r11           1.4347     0.5289   2.713 0.006677 **
r28         -18.1295     6.1484  -2.949 0.003191 **
r36           1.5231     0.3888   3.917 8.95e-05 ***
r3            5.4959     1.2741   4.313 1.61e-05 ***
r8           -3.3793     0.9457  -3.573 0.000353 ***
r32          16.5696     6.9327   2.390 0.016846 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1745.05  on 1259  degrees of freedom
Residual deviance:  666.58  on 1248  degrees of freedom
AIC: 690.58

Number of Fisher Scoring iterations: 8
```

We perform manually a likelihood-ratio-test-based forward selection (as described in Desbois) using only financial ratios as predictors.

```r
all <- glm(Y ~ ., data=data_afl %>% select(Y, starts_with('r')), family="binomial")

first_step <- add1(intercept_only, scope = formula(all), test = "LRT")
first_step <- first_step %>%
    tibble() %>%
```

```r
    add_column(variable=row.names(first_step)) %>%
    arrange(desc(LRT))
first_step
```

```
# A tibble: 23 x 6
      Df Deviance   AIC   LRT `Pr(>Chi)` variable
   <dbl>    <dbl> <dbl> <dbl>      <dbl> <chr>
 1     1     957.  961.  788.  2.27e-173 r1
 2     1     980.  984.  765.  1.95e-168 r14
 3     1     997. 1001.  748.  9.53e-165 r21
 4     1    1094. 1098.  651.  1.33e-143 r4
 5     1    1099. 1103.  646.  1.68e-142 r8
 6     1    1123. 1127.  622.  2.41e-137 r18
 7     1    1143. 1147.  602.  6.24e-133 r2
 8     1    1158. 1162.  587.  8.93e-130 r6
 9     1    1177. 1181.  568.  1.23e-125 r22
10     1    1185. 1189.  560.  9.00e-124 r30
# i 13 more rows
```

r1 is the most significant variable in the first step based on likelihood-ratio-test.

```r
second_step <- add1(update(intercept_only, ~. + r1), scope = formula(all), test = "LRT")
second_step <- second_step %>%
    tibble() %>%
    add_column(variable=row.names(second_step)) %>%
    arrange(desc(LRT))
second_step
```

```
# A tibble: 22 x 6
      Df Deviance   AIC   LRT `Pr(>Chi)` variable
   <dbl>    <dbl> <dbl> <dbl>      <dbl> <chr>
 1     1     804.  810. 153.    4.14e-35 r21
 2     1     809.  815. 148.    3.85e-34 r14
 3     1     827.  833. 130.    4.88e-30 r32
 4     1     850.  856. 107.    3.74e-25 r30
 5     1     860.  866.  97.5   5.26e-23 r8
 6     1     865.  871.  92.0   8.65e-22 r22
 7     1     868.  874.  89.2   3.59e-21 r18
 8     1     873.  879.  84.3   4.34e-20 r24
 9     1     880.  886.  77.6   1.26e-18 r12
```

```
10     1     882.  888.  75.3   4.12e-18 r28
# i 12 more rows
```

```
second_step <- add1(update(intercept_only, ~. + r1), scope = formula(all), test = "Rao")
second_step <- second_step %>%
    tibble() %>%
    add_column(variable=row.names(second_step)) %>%
    arrange(desc(`Rao score`))
second_step
```

```
# A tibble: 22 x 6
      Df Deviance   AIC `Rao score` `Pr(>Chi)` variable
   <dbl>    <dbl> <dbl>       <dbl>      <dbl> <chr>
 1     1     827.  833.       125.    4.55e-29 r32
 2     1     809.  815.       123.    1.56e-28 r14
 3     1     850.  856.        95.3   1.62e-22 r30
 4     1     860.  866.        84.0   5.02e-20 r8
 5     1     873.  879.        81.4   1.81e-19 r24
 6     1     868.  874.        78.4   8.32e-19 r18
 7     1     804.  810.        75.9   2.97e-18 r21
 8     1     880.  886.        75.1   4.42e-18 r12
 9     1     882.  888.        74.9   5.07e-18 r28
10     1     887.  893.        63.0   2.09e-15 r3
# i 12 more rows
```

r21 is the most significant variable in the second step (it differs from the article where r32 is selected (p70, the r32 in R is the same as in article (129.65) but r21 and r14 have higher LRT)).

```
third_step <- add1(update(intercept_only, ~. + r1 + r21), scope = formula(all), test = "LR
third_step <- third_step %>%
    tibble() %>%
    add_column(variable=row.names(third_step)) %>%
    arrange(desc(LRT))
third_step
```

```
# A tibble: 21 x 6
      Df Deviance   AIC   LRT `Pr(>Chi)` variable
   <dbl>    <dbl> <dbl> <dbl>      <dbl> <chr>
 1     1     714.  722.  90.2   2.17e-21 r14
```

```
 2     1     741.  749.  63.0   2.05e-15 r5
 3     1     744.  752.  60.5   7.20e-15 r2
 4     1     745.  753.  59.4   1.28e-14 r4
 5     1     746.  754.  58.3   2.26e-14 r3
 6     1     761.  769.  43.5   4.32e-11 r12
 7     1     764.  772.  40.2   2.34e-10 r11
 8     1     773.  781.  31.3   2.21e- 8 r8
 9     1     777.  785.  27.7   1.43e- 7 r7
10     1     793.  801.  11.5   7.03e- 4 r32
# i 11 more rows
```

r14 is the most significant variable in the third step.

```
fourth_step <- add1(update(intercept_only, ~. + r1 + r21 + r14), scope = formula(all), tes
fourth_step <- fourth_step %>%
    tibble() %>%
    add_column(variable=row.names(fourth_step)) %>%
    arrange(desc(LRT))
fourth_step
```

```
# A tibble: 20 x 6
      Df Deviance   AIC     LRT `Pr(>Chi)` variable
   <dbl>    <dbl> <dbl>   <dbl>      <dbl> <chr>
 1     1     711.  721.  3.25       0.0715 r17
 2     1     711.  721.  3.21       0.0733 r2
 3     1     711.  721.  2.98       0.0845 r11
 4     1     711.  721.  2.76       0.0969 r32
 5     1     712.  722.  2.37       0.124  r5
 6     1     712.  722.  2.28       0.131  r28
 7     1     712.  722.  2.11       0.146  r12
 8     1     712.  722.  1.85       0.174  r3
 9     1     713.  723.  1.55       0.213  r24
10     1     713.  723.  1.15       0.284  r4
11     1     713.  723.  0.903      0.342  r19
12     1     713.  723.  0.660      0.417  r8
13     1     713.  723.  0.607      0.436  r6
14     1     714.  724.  0.453      0.501  r30
15     1     714.  724.  0.370      0.543  r36
16     1     714.  724.  0.148      0.700  r22
17     1     714.  724.  0.145      0.703  r7
18     1     714.  724.  0.0622     0.803  r37
19     1     714.  724.  0.0260     0.872  r18
```

```
20    NA     714.  722. NA         NA      <none>
```

r17 is the most significant variable in the fourth step.

```
fifth_step <- add1(update(intercept_only, ~. + r1 + r21 + r14 + r17), scope = formula(all)
fifth_step <- fifth_step %>%
    tibble() %>%
    add_column(variable=row.names(fifth_step)) %>%
    arrange(desc(LRT))
fifth_step
```

```
# A tibble: 19 x 6
      Df Deviance   AIC      LRT `Pr(>Chi)` variable
   <dbl>    <dbl> <dbl>    <dbl>      <dbl> <chr>
 1     1     703.  715.  8.28       0.00401 r24
 2     1     704.  716.  6.72       0.00954 r28
 3     1     704.  716.  6.67       0.00982 r32
 4     1     705.  717.  5.72       0.0168  r11
 5     1     707.  719.  3.36       0.0669  r12
 6     1     708.  720.  3.28       0.0703  r2
 7     1     709.  721.  2.27       0.132   r5
 8     1     709.  721.  2.06       0.151   r3
 9     1     709.  721.  1.85       0.174   r18
10     1     709.  721.  1.82       0.177   r19
11     1     709.  721.  1.63       0.201   r30
12     1     709.  721.  1.52       0.218   r36
13     1     710.  722.  1.14       0.286   r4
14     1     710.  722.  0.786      0.375   r37
15     1     711.  723.  0.0927     0.761   r7
16     1     711.  723.  0.0445     0.833   r8
17     1     711.  723.  0.00413    0.949   r6
18     1     711.  723.  0.00389    0.950   r22
19    NA     711.  721. NA         NA       <none>
```

r24 is the most significant variable in the fifth step.

```
sixth_step <- add1(update(intercept_only, ~. + r1 + r21 + r14 + r17 + r24), scope = formul
sixth_step <- sixth_step %>%
    tibble() %>%
    add_column(variable=row.names(sixth_step)) %>%
    arrange(desc(LRT))
```

```r
sixth_step
```

```
# A tibble: 18 x 6
      Df Deviance   AIC     LRT `Pr(>Chi)` variable
   <dbl>    <dbl> <dbl>   <dbl>      <dbl> <chr>
 1     1     698.  712.  4.54       0.0331 r11
 2     1     698.  712.  4.33       0.0373 r2
 3     1     698.  712.  4.18       0.0410 r3
 4     1     699.  713.  3.71       0.0539 r28
 5     1     700.  714.  3.05       0.0805 r18
 6     1     700.  714.  3.03       0.0818 r32
 7     1     700.  714.  2.95       0.0857 r19
 8     1     700.  714.  2.85       0.0912 r5
 9     1     700.  714.  2.84       0.0917 r12
10     1     701.  715.  1.57       0.210  r4
11     1     701.  715.  1.35       0.246  r6
12     1     701.  715.  1.09       0.297  r37
13     1     702.  716.  0.913      0.339  r8
14     1     702.  716.  0.453      0.501  r7
15     1     702.  716.  0.0927     0.761  r30
16     1     703.  717.  0.0338     0.854  r22
17     1     703.  717.  0.0324     0.857  r36
18    NA     703.  715. NA         NA      <none>
```

`r11` is the most significant variable in the sixth step.

Equivalently the `step` routine produces the same results:

```r
# perform forward stepwise regression
forward_aic <- step(intercept_only, direction='forward', test = 'LRT', scope=formula(all),
summary(forward_aic)
```

```
Call:
glm(formula = Y ~ r1 + r21 + r14 + r17 + r24 + r11 + r28 + r36 +
    r3 + r8 + r32, family = "binomial", data = data_afl)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -11.1573     1.3883  -8.036 9.25e-16 ***
r1           10.3995     1.1853   8.773  < 2e-16 ***
r21           3.9664     1.5931   2.490 0.012787 *
```

```
r14           2.4414     0.6955   3.510 0.000448 ***
r17          36.0464     7.2134   4.997 5.82e-07 ***
r24          -4.1232     2.8366  -1.454 0.146058
r11           1.4347     0.5289   2.713 0.006677 **
r28         -18.1295     6.1484  -2.949 0.003191 **
r36           1.5231     0.3888   3.917 8.95e-05 ***
r3            5.4959     1.2741   4.313 1.61e-05 ***
r8           -3.3793     0.9457  -3.573 0.000353 ***
r32          16.5696     6.9327   2.390 0.016846 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1745.05  on 1259  degrees of freedom
Residual deviance:  666.58  on 1248  degrees of freedom
AIC: 690.58

Number of Fisher Scoring iterations: 8
```

```r
# perform forward stepwise regression
stepwisw_aic <- step(intercept_only, direction='both', test = 'LRT', scope=formula(all), k
summary(forward_aic)
```

```
Call:
glm(formula = Y ~ r1 + r21 + r14 + r17 + r24 + r11 + r28 + r36 +
    r3 + r8 + r32, family = "binomial", data = data_afl)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept) -11.1573     1.3883  -8.036 9.25e-16 ***
r1           10.3995     1.1853   8.773  < 2e-16 ***
r21           3.9664     1.5931   2.490 0.012787 *
r14           2.4414     0.6955   3.510 0.000448 ***
r17          36.0464     7.2134   4.997 5.82e-07 ***
r24          -4.1232     2.8366  -1.454 0.146058
r11           1.4347     0.5289   2.713 0.006677 **
r28         -18.1295     6.1484  -2.949 0.003191 **
r36           1.5231     0.3888   3.917 8.95e-05 ***
r3            5.4959     1.2741   4.313 1.61e-05 ***
r8           -3.3793     0.9457  -3.573 0.000353 ***
```

```
r32          16.5696     6.9327   2.390 0.016846 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1745.05  on 1259  degrees of freedom
Residual deviance:  666.58  on 1248  degrees of freedom
AIC: 690.58

Number of Fisher Scoring iterations: 8
```

We now perform manual likelihood-ratio-test-based forward selection with override when needed to match the procedure of Desbois article (i.e. forcing the selection of the same variables as in Desbois if `R` procedure disagrees).

```r
first_step_article <- add1(intercept_only, scope = formula(all), test = "LRT")
first_step_article <- first_step_article %>%
    tibble() %>%
    add_column(variable=row.names(first_step_article)) %>%
    arrange(desc(LRT))
head(first_step, 4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC   LRT `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl> <dbl>      <dbl> <chr>
1     1     957.  961.  788.  2.27e-173 r1
2     1     980.  984.  765.  1.95e-168 r14
3     1     997. 1001.  748.  9.53e-165 r21
4     1    1094. 1098.  651.  1.33e-143 r4
```

`r1` is the most significant variable in the first step.

```r
second_step_article <- add1(update(intercept_only, ~. + r1), scope = formula(all), test =
second_step_article <- second_step_article %>%
    tibble() %>%
    add_column(variable=row.names(second_step_article)) %>%
    arrange(desc(LRT))
head(second_step,4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC `Rao score` `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl>       <dbl>      <dbl> <chr>
1     1     827.  833.       125.    4.55e-29 r32
2     1     809.  815.       123.    1.56e-28 r14
3     1     850.  856.        95.3   1.62e-22 r30
4     1     860.  866.        84.0   5.02e-20 r8
```

In the article `r32` is the most significant variable in the second step: we force the selection of `r32` instead of `r21` proposed by `R`. We have exactly the same LRT as in the article:

```
third_step <- add1(update(intercept_only, ~. + r1 + r32), scope = formula(all), test = "LR
third_step <- third_step %>%
    tibble() %>%
    add_column(variable=row.names(third_step)) %>%
    arrange(desc(LRT))
head(third_step, 4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC   LRT `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl> <dbl>      <dbl> <chr>
1     1     741.  749.  86.6   1.31e-20 r14
2     1     789.  797.  38.7   4.93e-10 r17
3     1     790.  798.  37.9   7.31e-10 r8
4     1     790.  798.  37.1   1.10e- 9 r18
```

Like in the article `r14` is the most significant variable in the third step (with same LRT).

```
fourth_step <- add1(update(intercept_only, ~. + r1 + r32 + r14), scope = formula(all), tes
fourth_step <- fourth_step %>%
    tibble() %>%
    add_column(variable=row.names(fourth_step)) %>%
    arrange(desc(LRT))
head(fourth_step, 4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC   LRT   `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl> <dbl>        <dbl> <chr>
1     1     711.  721.  29.8 0.0000000482 r18
2     1     711.  721.  29.5 0.0000000556 r21
```

```
3     1     712.  722.  29.2 0.0000000648 r17
4     1     715.  725.  25.6 0.000000419  r28
```

In the article `r17` is the most significant variable in the fourth step: : we choose `r17`instead of `r18`.

```
fifth_step <- add1(update(intercept_only, ~. + r1 + r32 + r14 + r17), scope = formula(all)
fifth_step <- fifth_step %>%
    tibble() %>%
    add_column(variable=row.names(fifth_step)) %>%
    arrange(desc(LRT))
head(fifth_step,4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC   LRT `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl> <dbl>      <dbl> <chr>
1     1     698.  710.  13.7   0.000215 r36
2     1     699.  711.  12.9   0.000334 r37
3     1     700.  712.  11.9   0.000548 r12
4     1     701.  713.  11.1   0.000866 r11
```

Like in the article `r36` is the most significant variable in the fifth step (with same LRT)

```
sixth_step <- add1(update(intercept_only, ~. + r1 + r32 + r14 + r17 + r36), scope = formul
sixth_step <- sixth_step %>%
    tibble() %>%
    add_column(variable=row.names(sixth_step)) %>%
    arrange(desc(LRT))
head(sixth_step, 4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC   LRT `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl> <dbl>      <dbl> <chr>
1     1     689.  703.  9.08    0.00258 r12
2     1     689.  703.  8.53    0.00348 r3
3     1     690.  704.  8.41    0.00374 r11
4     1     691.  705.  7.36    0.00667 r2
```

Like in the article `r12` is the most significant variable in the sixth step (with same LRT). The routine (`SPSS`) in the article stops with this six variables.

Table 13. Results of the forward stepwise selection according to the likelihood ratio criterion: Stepwise summary (a)

Step	Improvement			Model			% correctly classified	Variable
	Chi-square	dof	Signif.	Chi-square	dof	Signif.		
1	787.930	1	.000	787.930	1	.000	82.2	IN: r1
2	129.653	1	.000	917.583	2	.000	85.2	IN: r32
3	86.621	1	.000	1004.204	3	.000	88.1	IN: r14
4	29.213	1	.000	1033.417	4	.000	88.6	IN: r17
5	13.693	1	.000	1047.110	5	.000	88.7	IN: r36
6	9.083	1	.003	1056.193	6	.000	89.1	IN: r12

Note: No other variable can be suppressed or included in the current model.

Looking at the Appendix of Desbois and `SPSS` documentation, it seems that the `SPSS` procedure `FSTEP` used by Desbois for the study performs a Rao score test instead of LRT so select the variables to entry the model (with a p-value threshold at 1%):

```
* Forward selection procedure to compute Binary Logistic Regression .
LOGISTIC REGRESSION VARIABLES diff
 /METHOD = FSTEP(LR) r1 r2 r3 r4 r5 r6 r7 r8 r11 r12 r14 r17
r18 r19 r21 r22
 r24 r28 r30 r32 r36 r37
 /SAVE = PRED PGROUP COOK LEVER DFBETA RESID
LRESID SRESID ZRESID DEV
 /CLASSPLOT /CASEWISE OUTLIER(2)
 /PRINT = GOODFIT CORR ITER(1) SUMMARY CI(95)
 /CRITERIA = PIN(.05) POUT(.10) ITERATE(20) CUT(.5) .
```

Contrarily to what is claimed in the article:

We now perform a manual (Rao)-score-test-based forward selection, and retrieve the variables selected by Desbois routine:

SPSS Statistics Subscription - Classic

ENTER. *Forced entry.* All variables are entered in a single step. This setting is the default if the NLMOD subcommand is omitted.

FSTEP. *Forward stepwise.* The variables (or interaction terms) that are specified on FSTEP are tested for entry into the model one by one, based on the significance level of the score statistic. The variable with the smallest significance less than PIN is entered into the model. After each entry, variables that are already in the model are tested for possible removal, based on the significance of the conditional statistic, the Wald statistic, or the likelihood-ratio criterion. The variable with the largest probability greater than the specified POUT value is removed, and the model is reestimated. Variables in the model are then evaluated again for removal. When no more variables satisfy the removal criterion, covariates that are not in the model are evaluated for entry. Model building stops when no more variables meet entry or removal criteria or when the current model is the same as a previous model.

```
first_step_article <- add1(intercept_only, scope = formula(all), test = "Rao")
first_step_article <- first_step_article %>%
    tibble() %>%
    add_column(variable=row.names(first_step_article)) %>%
    arrange(desc(`Rao score`))
head(first_step, 4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC   LRT `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl> <dbl>      <dbl> <chr>
1     1     957.  961.  788.  2.27e-173 r1
2     1     980.  984.  765.  1.95e-168 r14
3     1     997. 1001.  748.  9.53e-165 r21
4     1    1094. 1098.  651.  1.33e-143 r4
```

`r1` is the most significant variable in the first step.

```
second_step_article <- add1(update(intercept_only, ~. + r1), scope = formula(all), test =
second_step_article <- second_step_article %>%
    tibble() %>%
    add_column(variable=row.names(second_step_article)) %>%
    arrange(desc(`Rao score`))
head(second_step,4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC `Rao score` `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl>       <dbl>      <dbl> <chr>
1     1     827.  833.       125.    4.55e-29 r32
2     1     809.  815.       123.    1.56e-28 r14
3     1     850.  856.        95.3   1.62e-22 r30
4     1     860.  866.        84.0   5.02e-20 r8
```

In the article `r32` is the most significant variable in the second step.

Selection of regressors: the forward stepwise procedure

The stepwise procedure of forward selection for the regressors is similar to that used in discriminating analysis: with each step *k*, the variable X_k is selected with the smallest p-value, provided that this p-value is not higher than a threshold value a priori fixed (here 1%). The principle of the test used to select an additional regressor is based on the deflation of deviance induced by the likelihood ratio (LR). This ratio is obtained by dividing the probability of the incomplete model (at the step *k-1*) by the probability of the complete model (step *k*):

$$G_k = -2\log(LR) = -2\log\left[\frac{Likelihood\ without\ X_k}{Likelihood\ with\ X_k}\right].$$

Under the null hypothesis *H0* of nullity of the logistic regression parameter β_k, the quantity $-2*\log(LR)$ follows a distribution of Chi-square $\chi^2(\nu)$ with ν degrees of freedom, provided that the sample size is sufficiently large. The number of degrees of freedom ν is the difference between the number of parameters of the complete model and the number of parameters of the incomplete model (here $\nu = 1$ at each step). For each included variable, the chi-squared test leads to a rejection of the null hypothesis with a p-value lower than 1 out of ten thousand for the first 5 steps and of 3 out of thousand in the last step (Table 13).

```r
third_step <- add1(update(intercept_only, ~. + r1 + r32), scope = formula(all), test = "Ra
third_step <- third_step %>%
    tibble() %>%
    add_column(variable=row.names(third_step)) %>%
    arrange(desc(`Rao score`))
head(third_step, 4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC `Rao score` `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl>       <dbl>      <dbl> <chr>
1     1     741.  749.        71.1   3.37e-17 r14
2     1     789.  797.        38.0   7.04e-10 r17
3     1     790.  798.        35.1   3.06e- 9 r18
4     1     790.  798.        34.4   4.52e- 9 r8
```

Like in the article `r14` is the most significant variable in the third step.

```r
fourth_step <- add1(update(intercept_only, ~. + r1 + r32 + r14), scope = formula(all), tes
fourth_step <- fourth_step %>%
    tibble() %>%
    add_column(variable=row.names(fourth_step)) %>%
    arrange(desc(`Rao score`))
head(fourth_step, 4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC `Rao score`   `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl>       <dbl>        <dbl> <chr>
1     1     712.  722.        28.6 0.0000000872 r17
2     1     711.  721.        27.9 0.000000127  r18
3     1     715.  725.        23.8 0.00000108   r28
4     1     711.  721.        18.8 0.0000141    r21
```

In the article `r17` is the most significant variable in the fourth step.

```r
fifth_step <- add1(update(intercept_only, ~. + r1 + r32 + r14 + r17), scope = formula(all)
fifth_step <- fifth_step %>%
    tibble() %>%
    add_column(variable=row.names(fifth_step)) %>%
    arrange(desc(`Rao score`))
head(fifth_step,4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC `Rao score` `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl>       <dbl>      <dbl> <chr>
1     1     698.  710.        13.6   0.000230 r36
2     1     700.  712.        12.4   0.000438 r12
3     1     699.  711.        12.2   0.000490 r37
4     1     701.  713.        11.8   0.000593 r11
```

Like in the article `r36` is the most significant variable in the fifth step.

```
sixth_step <- add1(update(intercept_only, ~. + r1 + r32 + r14 + r17 + r36), scope = formul
sixth_step <- sixth_step %>%
    tibble() %>%
    add_column(variable=row.names(sixth_step)) %>%
    arrange(desc(`Rao score`))
head(sixth_step, 4)
```

```
# A tibble: 4 x 6
     Df Deviance   AIC `Rao score` `Pr(>Chi)` variable
  <dbl>    <dbl> <dbl>       <dbl>      <dbl> <chr>
1     1     689.  703.        9.16    0.00248 r12
2     1     689.  703.        8.70    0.00319 r3
3     1     690.  704.        8.51    0.00354 r11
4     1     691.  705.        6.82    0.00903 r2
```

Like in the article `r12` is the most significant variable in the sixth step. The routine (`SPSS`) in the article stops with this six variables.

So basically the article prints and focus on a statistic (LRT, Table 13) which is different form the one used by the effective procedure (Rao score test) to perform variable selection, hence the difference with our findings. Also the AIC criterion (based on LRT) implies a higher p-value than the one used by Desbois to stop the procedure, hence more variables are selected by `stats::step()` than Desbois procedure.

R performs automated AIC/BIC based variable selections using (step)

```
forward_bic <- step(intercept_only, direction='forward', test = 'LRT', scope=formula(all),
summary(forward_bic)
```

```
Call:
glm(formula = Y ~ r1 + r21 + r14, family = "binomial", data = data_afl)
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.3243     0.3196 -16.657  < 2e-16 ***
r1            3.9493     0.5754   6.864 6.70e-12 ***
r21           7.3123     0.8965   8.157 3.44e-16 ***
r14           2.5822     0.3113   8.294  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1745.05  on 1259  degrees of freedom
Residual deviance:  714.09  on 1256  degrees of freedom
AIC: 722.09

Number of Fisher Scoring iterations: 7
```

```r
forward_aic <- step(intercept_only, direction='forward', scope=formula(all), k=9.21, trace
summary(forward_aic)
```

```
Call:
glm(formula = Y ~ r1 + r21 + r14, family = "binomial", data = data_afl)

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -5.3243     0.3196 -16.657  < 2e-16 ***
r1            3.9493     0.5754   6.864 6.70e-12 ***
r21           7.3123     0.8965   8.157 3.44e-16 ***
r14           2.5822     0.3113   8.294  < 2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1745.05  on 1259  degrees of freedom
Residual deviance:  714.09  on 1256  degrees of freedom
AIC: 722.09

Number of Fisher Scoring iterations: 7
```

```r
backward_aic <- step(all, direction='backward', test = 'LRT', scope=formula(all), k=2, tra
summary(backward_aic)
```

```
Call:
glm(formula = Y ~ r1 + r3 + r8 + r12 + r14 + r17 + r22 + r28 +
    r30 + r36 + r37, family = "binomial", data = data_afl %>%
    select(Y, starts_with("r")))

Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -9.9854     1.5001  -6.657 2.80e-11 ***
r1           10.2464     1.0700   9.576  < 2e-16 ***
r3            5.3125     1.2413   4.280 1.87e-05 ***
r8           -3.2304     0.9246  -3.494 0.000476 ***
r12           0.9784     0.3746   2.612 0.009001 **
r14           2.5304     0.7107   3.560 0.000371 ***
r17          34.7873     5.3459   6.507 7.65e-11 ***
r22           1.8935     0.8032   2.357 0.018404 *
r28         -12.0852     2.4397  -4.954 7.28e-07 ***
r30           8.0662     3.1521   2.559 0.010497 *
r36           1.4516     0.3629   4.000 6.34e-05 ***
r37          -1.4783     0.9176  -1.611 0.107190
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1745.05  on 1259  degrees of freedom
Residual deviance:  663.66  on 1248  degrees of freedom
AIC: 687.66

Number of Fisher Scoring iterations: 8
```

```r
backward_bic <- step(all, direction='backward', test = 'LRT', scope=formula(all), k=log(nr
summary(backward_bic)
```

```
Call:
glm(formula = Y ~ r1 + r3 + r8 + r12 + r14 + r17 + r28 + r36,
    family = "binomial", data = data_afl %>% select(Y, starts_with("r")))
```

```
Coefficients:
            Estimate Std. Error z value Pr(>|z|)
(Intercept)  -9.3068     1.0442  -8.913  < 2e-16 ***
r1            9.1338     0.9539   9.575  < 2e-16 ***
r3            4.0983     1.1739   3.491 0.000481 ***
r8           -2.2337     0.8121  -2.751 0.005949 **
r12           1.0991     0.3639   3.020 0.002528 **
r14           2.5999     0.7224   3.599 0.000319 ***
r17          31.9448     4.4733   7.141 9.26e-13 ***
r28          -7.8125     1.2101  -6.456 1.08e-10 ***
r36           1.5837     0.2917   5.430 5.65e-08 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1745.05  on 1259  degrees of freedom
Residual deviance:  673.16  on 1251  degrees of freedom
AIC: 691.16

Number of Fisher Scoring iterations: 7
```

```r
#define intercept-only model
intercept_only_w <- glm(Y ~ 1, data=don_desbois_winsorized, family="binomial")

#define model with all predictors
all_w <- glm(Y ~ ., data=don_desbois_winsorized, family="binomial")

#perform forward stepwise regression AIC
forward_aic_w <- step(intercept_only_w,
                      direction='forward',
                      scope=formula(all_w),
                      k=2,
                      trace = FALSE)

#perform forward stepwise regression BIC
forward_bic_w <- step(intercept_only_w,
                      direction='forward',
                      scope=formula(all_w),
                      k=log(nrow(don_desbois_winsorized)),
                      trace = FALSE)
```

```r
summary(forward_aic_w)
```

```
Call:
glm(formula = Y ~ r1 + r21 + r14 + HECTARE + CNTY + r5 + STATUS +
    r7 + r6 + AGE + r37 + OWNLAND + r17 + r18 + ToF + r3, family = "binomial",
    data = don_desbois_winsorized)

Coefficients:
                                          Estimate Std. Error z value Pr(>|z|)
(Intercept)                              -4.479203   1.673417  -2.677 0.007436
r1                                       19.699710   3.477155   5.665 1.47e-08
r21                                       7.091515   1.674065   4.236 2.27e-05
r14                                       2.039834   0.565470   3.607 0.000309
HECTARE                                  -0.035654   0.004401  -8.102 5.40e-16
CNTYNord                                 -2.706287   0.435148  -6.219 5.00e-10
CNTYOrne                                 -2.520852   0.493223  -5.111 3.20e-07
CNTYSeine-Maritime                       -0.075642   0.384078  -0.197 0.843872
r5                                      -14.636807   4.246808  -3.447 0.000568
STATUSEntreprise individuelle            -1.287226   0.379712  -3.390 0.000699
r7                                        9.504055   2.028865   4.684 2.81e-06
r6                                       -5.094662   1.698924  -2.999 0.002711
AGE                                       0.047111   0.016565   2.844 0.004455
r37                                      -5.358372   1.632771  -3.282 0.001032
OWNLANDyes                               -0.729304   0.305731  -2.385 0.017059
r17                                      46.935358  12.204133   3.846 0.000120
r18                                     -35.548694  11.641241  -3.054 0.002260
ToFgeneral cropping                      -1.128581   0.381741  -2.956 0.003112
ToFdairy farm                            -0.258623   0.323345  -0.800 0.423805
ToFmixed livestock                       -0.990716   0.626063  -1.582 0.113546
ToFvarious crops and livestock combined  -0.266008   0.768993  -0.346 0.729405
ToFsoilless breeding                      0.973365   1.203540   0.809 0.418658
r3                                        2.998904   1.735381   1.728 0.083971

(Intercept)                             **
r1                                      ***
r21                                     ***
r14                                     ***
HECTARE                                 ***
CNTYNord                                ***
CNTYOrne                                ***
CNTYSeine-Maritime
r5                                      ***
```

```
STATUSEntreprise individuelle           ***
r7                                      ***
r6                                      **
AGE                                     **
r37                                     **
OWNLANDyes                              *
r17                                     ***
r18                                     **
ToFgeneral cropping                     **
ToFdairy farm
ToFmixed livestock
ToFvarious crops and livestock combined
ToFsoilless breeding
r3                                      .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

    Null deviance: 1745.1  on 1259  degrees of freedom
Residual deviance:  508.9  on 1237  degrees of freedom
AIC: 554.9

Number of Fisher Scoring iterations: 7
```

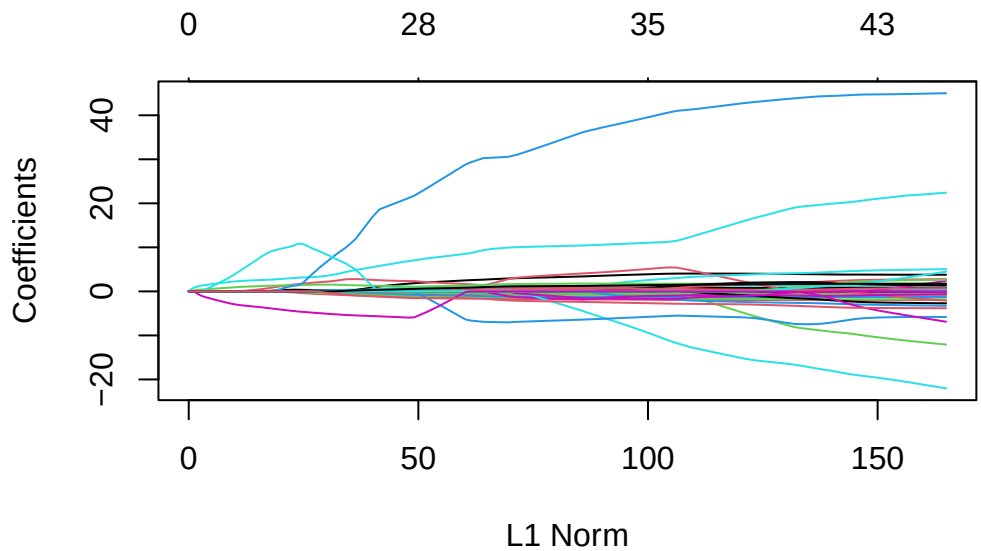
9.5 Penalized Logistic Regression

Perform penalized logistic regression with the Desbois data set. You can find help here.

```r
# YOUR CODE HERE
```

For example below we show the lasso "path" of coefficients when increasing λ (from right to left).

```r
Y <- don_desbois %>% pull(Y)
desbois_lasso <- glmnetUtils::glmnet(Y ~ ., data=don_desbois, family="binomial", alpha=1)

plot(desbois_lasso)
```



We can extract for each λ the value of coefficients:

```
lasso_result <- as_tibble(as.matrix(cbind(desbois_lasso$lambda, t(desbois_lasso$beta))))
names(lasso_result) <-  c("lambda", row.names(desbois_lasso$beta))
lasso_result
```

```
# A tibble: 100 x 46
   lambda CNTYEure CNTYNord CNTYOrne `CNTYSeine-Maritime` STATUScompany
    <dbl>    <dbl>    <dbl>    <dbl>                 <dbl>         <dbl>
 1  0.324        0        0        0                     0             0
 2  0.295        0        0        0                     0             0
 3  0.269        0        0        0                     0             0
 4  0.245        0        0        0                     0             0
 5  0.223        0        0        0                     0             0
 6  0.203        0        0        0                     0             0
 7  0.185        0        0        0                     0             0
 8  0.169        0        0        0                     0             0
 9  0.154        0        0        0                     0             0
10  0.140        0        0        0                     0             0
# i 90 more rows
# i 40 more variables: `STATUSEntreprise individuelle` <dbl>, HECTARE <dbl>,
#   ToFcereals <dbl>, `ToFgeneral cropping` <dbl>, `ToFdairy farm` <dbl>,
```

```
#   `ToFmixed livestock` <dbl>,
#   `ToFvarious crops and livestock combined` <dbl>,
#   `ToFsoilless breeding` <dbl>, OWNLANDno <dbl>, OWNLANDyes <dbl>, AGE <dbl>,
#   HARVEST1988 <dbl>, HARVEST1989 <dbl>, HARVEST1990 <dbl>, ...
```

`glmnet` also provides an automatic and efficient procedure to select λ implementing K-fold Cross-Validation (see next question for an example implementation of the K-fold Cross-Validation method in general).

Functions `glmnet::cv.glmnet` / `glmnetUtils::cv.glmnet` fit the lasso path (ie multiple penalized models for multiple values of lambda) with the selection of the best lambda by K-Fold Cross-Validation (by default 10-fold) using a given criterion such as the AUC (`type.measure = "auc"`).

The function computes two optimal values: `lambda.min` is the value of lambda that gives minimum mean Cross-validated criterion; `lambda.1se` is the value of lambda that gives the most regularized model (highest lambda) such that the Cross-validated criterion is within one standard error of the minimum, it favours penalization/parsimony versus `lambda.min` (see here).

```r
# glmnet::cv.glmnet / glmnetUtils::cv.glmnet fit the lasso path (ie multiple penalized mod
# with selection of the best lambda by cross-validation (by default 10-fold)
# using a given criterion (type.measure = "auc")
desbois_lasso_cv <- glmnetUtils::cv.glmnet(Y ~ ., data=don_desbois, family="binomial", alp


# We can extract model coefficients for lambda = lambda.1se
lasso_coeffs <- coef(desbois_lasso_cv, s = "lambda.1se")
data.frame(name = lasso_coeffs@Dimnames[[1]][lasso_coeffs@i + 1], coefficient = lasso_coef
```

```
                           name   coefficient
1                   (Intercept) -2.127806e+00
2                      CNTYEure  2.348695e-01
3                      CNTYNord -8.965654e-01
4                      CNTYOrne -6.263470e-01
5                 STATUScompany  4.200001e-01
6  STATUSEntreprise individuelle -6.325638e-14
7                       HECTARE -2.123148e-02
8            ToFgeneral cropping -4.072887e-01
9             ToFmixed livestock -2.301024e-01
10                     OWNLANDno  4.605143e-02
11                    OWNLANDyes -6.891329e-15
12                           AGE  7.445191e-03
```

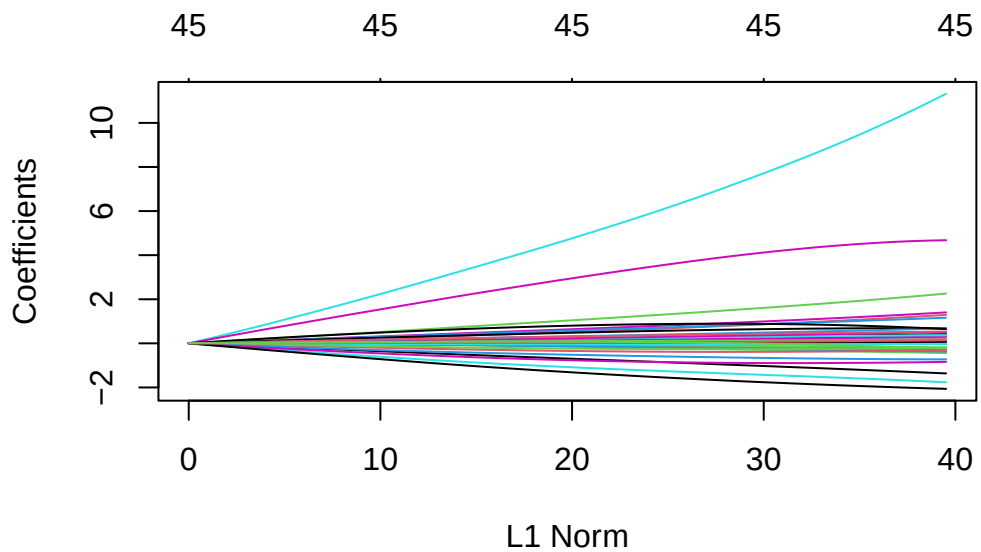
```
13                    HARVEST1988  1.517231e-01
14                    HARVEST1989  6.618315e-02
15                    HARVEST1992 -1.511825e-01
16                    HARVEST1993 -3.970665e-02
17                             r1  4.857597e+00
18                             r3  3.018614e-01
19                             r4  2.791796e+00
20                             r6  1.295167e-03
21                             r7  1.934853e-01
22                            r14  1.384477e+00
23                            r17  1.181177e+01
24                            r18  5.010237e+00
25                            r32 -5.457207e+00
26                            r36  2.151685e-01
27                            r37 -9.382449e-01
```

By default the `predict` function for a `cv.glmnet` model uses `lambda.1se`, but we can specify any value of lambda, in particular `lambda.min`, also note that the predict function outputs a `matrix` (`predict` functions in `R` outputs a vector in general):

```
tibble(one_se = as.vector(predict(desbois_lasso_cv, newdata = don_desbois, s = desbois_las
       min = as.vector(predict(desbois_lasso_cv, newdata = don_desbois, s = desbois_lasso_
       default = as.vector(predict(desbois_lasso_cv, newdata = don_desbois,  type = "respo
```

```
# A tibble: 1,260 x 3
    one_se     min default
     <dbl>   <dbl>   <dbl>
 1 0.0356  0.0111  0.0356
 2 0.0641  0.0234  0.0641
 3 0.0722  0.0211  0.0722
 4 0.0275  0.0111  0.0275
 5 0.0477  0.00953 0.0477
 6 0.0266  0.00949 0.0266
 7 0.196   0.179   0.196
 8 0.302   0.245   0.302
 9 0.00616 0.00262 0.00616
10 0.0674  0.0358  0.0674
# i 1,250 more rows
```

```
Y <- don_desbois %>% pull(Y)
desbois_ridge <- glmnetUtils::glmnet(Y ~ ., data=don_desbois, family="binomial", alpha=0)

plot(desbois_ridge)
```



9.6 Model assessment

The aim to find an optimal value of the lambda parameter for Lasso regression without relying on the built-in functions `glmnetUtils::cv.glmnet` or `glmnet::cv.glmnet`. It will allow to generalize to cases where standard K-fold Cross Validation procedures do not apply, as for example grouped or panel data.

First we assume a simpler problem where the data set is split in training and testing set (hold out).

We choose 70%/30% split of the data set (training and testing set). Here the data set is balanced and train/test split using a simple random sampling shows a similar distribution of default:

```
# Using base R
set.seed(1987)  # for reproducibility
index_1 <- sample(1:nrow(don_desbois), round(nrow(don_desbois) * 0.7))
```

```r
train_1 <- don_desbois[index_1, ]
test_1  <- don_desbois[-index_1, ]

# table(don_desbois$Y) %>% prop.table()
#        0        1
# 0.518254 0.481746

# table(train_1$Y) %>% prop.table()
#         0         1
# 0.5136054 0.4863946

# table(test_1$Y) %>% prop.table()
#         0         1
# 0.5291005 0.4708995

# Using rsample package
set.seed(1987)  # for reproducibility
split_1  <- rsample::initial_split(don_desbois, prop = 0.7)
train_2  <- rsample::training(split_1)
test_2   <- rsample::testing(split_1)

# table(train_2$Y) %>% prop.table()
#         0         1
# 0.5136054 0.4863946

# table(test_2$Y) %>% prop.table()
#         0         1
# 0.5291005 0.4708995
```

In case we want to avoid the split of an already imbalanced data set to further skew distributions we may want to use Stratified sampling:

```r
split_strat  <- rsample::initial_split(don_desbois, prop = 0.7,
                                       strata = "Y")
train_strat  <- rsample::training(split_strat)
test_strat   <- rsample::testing(split_strat)

# table(don_desbois$Y) %>% prop.table()
#        0        1
# 0.518254 0.481746
```

```
# table(train_strat$Y) %>% prop.table()
#         0         1
# 0.5187287 0.4812713

# table(test_1$Y) %>% prop.table()
#         0         1
# 0.5171504 0.4828496
```

We fit the Lasso path (more details here) on the training set making use of `glmnetUtils::glmnet` or `glmnet::glmnet` functions

We use ratios from `Desbois` and fit the Lasso path using `glmnetUtils::glmnet`, we `tidy` the result in a data set `lasso_result` showing coefficients w.r.t. lambda:

```
# YOUR CODE HERE
default_lasso <- glmnetUtils::glmnet(Y ~ .,
                                     data=train_strat %>% select(Y, r1:r37),
                                     family="binomial",
                                     alpha=1,
                                     standardize = TRUE)

lasso_result <- as_tibble(as.matrix(cbind(default_lasso$lambda, default_lasso$a0, t(as.mat
names(lasso_result) <-  c("lambda", "(Intercept)", row.names(default_lasso$beta))
lasso_result
```

```
# A tibble: 96 x 24
   lambda `(Intercept)`    r1    r2    r3    r4    r5    r6    r7    r8   r11
    <dbl>         <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  0.326       -0.0749 0         0     0     0     0     0     0     0     0
 2  0.297       -0.284  0.356     0     0     0     0     0     0     0     0
 3  0.271       -0.482  0.694     0     0     0     0     0     0     0     0
 4  0.247       -0.662  0.896     0     0     0     0     0     0     0     0
 5  0.225       -0.724  1.05      0     0     0     0     0     0     0     0
 6  0.205       -0.761  1.20      0     0     0     0     0     0     0     0
 7  0.187       -0.800  1.34      0     0     0     0     0     0     0     0
 8  0.170       -0.848  1.47      0     0     0     0     0     0     0     0
 9  0.155       -0.913  1.57      0     0     0     0     0     0     0     0
10  0.141       -0.978  1.67      0     0     0     0     0     0     0     0
# i 86 more rows
# i 13 more variables: r12 <dbl>, r14 <dbl>, r17 <dbl>, r18 <dbl>, r19 <dbl>,
#   r21 <dbl>, r22 <dbl>, r24 <dbl>, r28 <dbl>, r30 <dbl>, r32 <dbl>,
#   r36 <dbl>, r37 <dbl>
```

We use the `glmnet` object and `predict` function, for each lambda of the Lasso path (or better for all lambdas at once) to obtain the predicted probabilities on the testing/holdout set.

The `predict` function from `glmnet` allows to predict the whole Lasso path on a testing set, we `tidy` the result in a data set `predict_test` showing predicted probabilities on testing set w.r.t. lambda:

```
# YOUR CODE HERE
predict_test <- as_tibble(predict(default_lasso,
                                  newdata = test_strat %>% select(r1:r37),
                                  s = default_lasso$lambda,
                                  type ="response"))
names(predict_test) <-  glue::glue('Lambda {round(default_lasso$lambda,6)}')
predict_test
```

```
# A tibble: 379 x 96
   `Lambda 0.326055` `Lambda 0.297089` `Lambda 0.270697` `Lambda 0.246649`
               <dbl>             <dbl>             <dbl>             <dbl>
 1             0.481             0.456             0.433             0.408
 2             0.481             0.493             0.504             0.499
 3             0.481             0.460             0.441             0.425
 4             0.481             0.473             0.466             0.454
 5             0.481             0.441             0.403             0.369
 6             0.481             0.468             0.456             0.439
 7             0.481             0.488             0.495             0.490
 8             0.481             0.500             0.519             0.530
 9             0.481             0.451             0.422             0.392
10             0.481             0.495             0.508             0.510
# i 369 more rows
# i 92 more variables: `Lambda 0.224737` <dbl>, `Lambda 0.204772` <dbl>,
#   `Lambda 0.186581` <dbl>, `Lambda 0.170005` <dbl>, `Lambda 0.154903` <dbl>,
#   `Lambda 0.141141` <dbl>, `Lambda 0.128603` <dbl>, `Lambda 0.117178` <dbl>,
#   `Lambda 0.106768` <dbl>, `Lambda 0.097283` <dbl>, `Lambda 0.088641` <dbl>,
#   `Lambda 0.080766` <dbl>, `Lambda 0.073591` <dbl>, `Lambda 0.067054` <dbl>,
#   `Lambda 0.061097` <dbl>, `Lambda 0.055669` <dbl>, ...
```

Using the preceding step we are be able to compute the hold-out error (or criterion if AUC) for each lambda (with a `for` loop), and then conclude on the best λ:

```
# YOUR CODE HERE
res_auc_lasso <- list()
```

```r
for(l in seq_len(length(default_lasso$lambda))){
  pred <- ROCR::prediction(predict_test[,l] %>% pull(), test_strat$Y)
  # compute empirical AUC
  auc <- ROCR::performance(pred, measure = "auc")
  auc <- auc@y.values[[1]]
  res_auc_lasso[[l]] <- tibble(lambda = default_lasso$lambda[l], auc = auc)

}
res_auc_lasso <- bind_rows(res_auc_lasso)
(res_auc_lasso_max <- res_auc_lasso %>% filter(auc == max(auc)))
```

```
# A tibble: 1 x 2
   lambda   auc
    <dbl> <dbl>
1 0.00215 0.960
```

Showing the results for the lambda selection using holdout set:

```r
glimpse(lasso_result %>% filter(lambda ==res_auc_lasso_max %>% pull(lambda)))
```

```
Rows: 1
Columns: 24
$ lambda        <dbl> 0.002145225
$ `(Intercept)` <dbl> -6.4987
$ r1            <dbl> 6.371395
$ r2            <dbl> 0
$ r3            <dbl> 2.674514
$ r4            <dbl> 0
$ r5            <dbl> 0
$ r6            <dbl> 0
$ r7            <dbl> 0.5633372
$ r8            <dbl> -1.263007
$ r11           <dbl> 0.7573327
$ r12           <dbl> 0
$ r14           <dbl> 2.374612
$ r17           <dbl> 29.66679
$ r18           <dbl> 0
$ r19           <dbl> -0.3495623
$ r21           <dbl> 0.4139145
$ r22           <dbl> 0
```

```
$ r24           <dbl> 0
$ r28           <dbl> -6.670012
$ r30           <dbl> 0
$ r32           <dbl> 0
$ r36           <dbl> 0.697629
$ r37           <dbl> -0.522884
```

Going one step further, we replace the hold-out approach by a K-Fold Cross-Validation (for each fold of the data, we compute the cross-validation error (or criterion) for each lambda (we use two imbricated `for` loops)); we then conclude on the best lambda in terms of mean Cross-Validation error (or criterion) and are able to compare the lambda chosen with the built-in function `cv.glmnet` described here.

We implement below a manual selection of lambda using 10-Fold Cross-Validation:

```
# YOUR CODE HERE
set.seed(1987)

res_list <- list()
# we repeat the k-fold cross validation nb_iter times
nb_iter <- 1 # Usually 5 x 10 fold validation, set to 1 if too slow

# as the AUC is the metric requested, we define a function to compute the AUC given a Scor
# values Y
func_auc <- function(X,Y){
      pred <- ROCR::prediction(X, Y)
      auc <- ROCR::performance(pred, measure = "auc")
      auc <- auc@y.values[[1]]
}

# using the same auc function as glmnet (using survival::concordance)
# https://github.com/search?q=repo%3Acran%2Fglmnet%20concordance&type=code
func_auc_glmnet <- function(X,Y){
      survival::concordance(Y~X)$concordance
}


for (j in 1:nb_iter) {

# we split randomly the dataset into nb_blocks folds

     nb_blocks <- 10 # number of folds/blocks, set to 5 if too low
```

```r
blocks <- sample(rep(1:nb_blocks,nrow(train_strat))[1:nrow(train_strat)])

result <- data.frame(matrix(nrow=dim(train_strat), ncol=length(default_lasso$lambda))
indiv_auc <- list()

# we give names to the result columns corresponding to the lambdas assessed
names(result) <- glue::glue('Lambda {round(lasso_result$lambda,6)}')

for (i in 1:nb_blocks) {
      # print(i)

      # we sequentially use each fold as a testing set / the complement being used as
      XX_train <- train_strat[blocks!=i,]
      XX_test <- train_strat[blocks==i,]

      # we then fit different models we want to assess

      # lasso
      lasso <- glmnetUtils::glmnet(Y ~ .,
                                   data = XX_train %>% select(Y, r1:r37),
                                   family="binomial",
                                   alpha=1,
                                   lambda=default_lasso$lambda,
                                   # to allow comparability between folds, we use user-
                                   # for example those of the lasso fitted on the whole
                                   standardize = TRUE)

      # predicting probas on testing set / out of fold
      predict_test <- as_tibble(predict(lasso,
                              newdata = XX_test %>% select(r1:r37),
                              s = lasso$lambda,
                              type ="response"))

      pred = predict(lasso,
                     newdata=XX_test %>% select(r1:r37),
                     s = lasso$lambda,
                     type = "response")

      result[blocks==i,] = pred

      # computing AUC for each lambdas for the i-th fold
```

```r
            indiv_auc[[i]] = apply(result[blocks==i,], 2, func_auc_glmnet, Y=XX_test$Y)

      }



      # we compute a 'global' AUC for each of the lambdas
      res_list[[j]] = list(auc = apply(result, 2, func_auc, Y=train_strat$Y),
                           auc_glmnet = apply(result, 2, func_auc_glmnet, Y=train_strat$Y),
                           indiv_auc = indiv_auc,
                           result = result)
  }

  #

  res_for_plot_manual <- tibble(lambda = default_lasso$lambda,
                                # Global AUC using ROCR
                                auc = res_list[[1]]$auc,
                                # Global AUC using glmnet function
                                auc_glmnet = res_list[[1]]$auc_glmnet,
                                # Computing the mean of i-th fold AUC for each lambda
                                auc_indiv = bind_rows(res_list[[1]]$indiv_auc) %>% summarise

  # Showing Cross-validated lambda
  res_for_plot_manual %>% filter(auc == max(auc)|auc_glmnet == max(auc_glmnet)|auc_indiv ==
```

```
# A tibble: 2 x 4
   lambda   auc auc_glmnet auc_indiv
    <dbl> <dbl>      <dbl>     <dbl>
1 0.00215 0.949      0.949     0.950
2 0.00112 0.949      0.949     0.949
```

We do the same using `cv.glmnet` ensuring number of folds, chosen folds, and lambdas used are the same as before:

```r
  # See here for details on parameters used https://glmnet.stanford.edu/reference/cv.glmnet.

  # lambda
  # Optional user-supplied lambda sequence; default is NULL, and glmnet chooses its own sequ

  # foldid
```

```r
# an optional vector of values between 1 and nfolds identifying what fold each observation

# alignment
# This is an experimental argument, designed to fix the problems users were having with CV


default_lasso_cv <- glmnetUtils::cv.glmnet(Y ~ .,
                                           data = train_strat %>% select(Y, r1:r37),
                                           family="binomial",
                                           alpha=1,
                                           lambda = default_lasso$lambda, # force using same
                                           foldid = blocks, # using the same folds as before
                                           alignment = "lambda",
                                           standardize = TRUE,
                                           type.measure = "auc")

lasso_result_cv <- as_tibble(as.matrix(cbind(default_lasso_cv$glmnet.fit$lambda, default_l
names(lasso_result_cv) <-  c("lambda", "(Intercept)", row.names(default_lasso_cv$glmnet.fi


lasso_coeffs_1se <- coef(default_lasso_cv, s = "lambda.1se")
(lasso_coeffs_1se <- tibble(name = lasso_coeffs_1se@Dimnames[[1]][lasso_coeffs_1se@i + 1],
                            coefficient = lasso_coeffs_1se@x))
```

```
# A tibble: 5 x 2
  name        coefficient
  <chr>             <dbl>
1 (Intercept)      -1.04
2 r1                1.77
3 r14               0.526
4 r18               1.37
5 r32              -2.01
```

```r
lasso_coeffs_min <- coef(default_lasso_cv, s = "lambda.min")
(lasso_coeffs_min <- tibble(name = lasso_coeffs_min@Dimnames[[1]][lasso_coeffs_min@i + 1],
                            coefficient = lasso_coeffs_min@x))
```

```
# A tibble: 13 x 2
   name        coefficient
   <chr>             <dbl>
```

```
 1 (Intercept)      -6.50
 2 r1                6.37
 3 r3                2.67
 4 r7                0.563
 5 r8               -1.26
 6 r11               0.757
 7 r14               2.37
 8 r17              29.7
 9 r19              -0.350
10 r21               0.414
11 r28              -6.67
12 r36               0.698
13 r37              -0.523
```

```
res_for_plot_glmnet <- tibble(lambda = default_lasso_cv$lambda, auc = default_lasso_cv$cvm
res_for_plot_glmnet %>% filter(auc == max(auc))
```

```
# A tibble: 1 x 2
   lambda   auc
    <dbl> <dbl>
1 0.00215 0.949
```

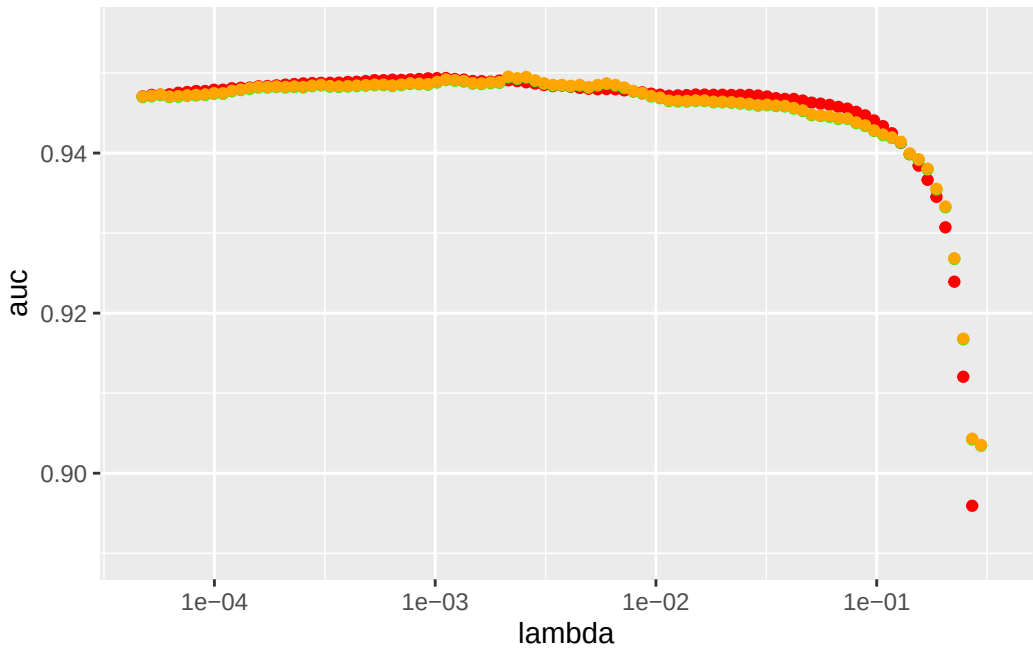
Cross-Validated lambda is the same as in our implementation. We also display below `lambda.1se`:

```
lasso_result_cv %>% filter(lambda %in% c(default_lasso_cv$lambda.1se, default_lasso_cv$lam
```

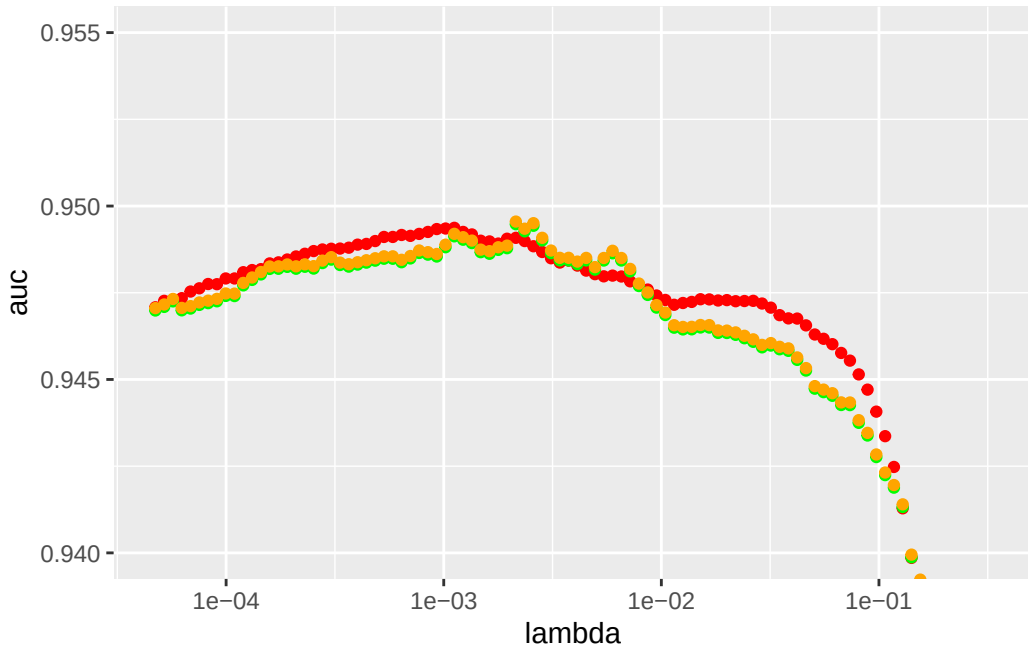
```
# A tibble: 2 x 24
   lambda `(Intercept)`    r1    r2    r3    r4    r5    r6    r7    r8   r11
    <dbl>         <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 0.129           -1.04  1.77     0  0        0     0     0 0      0    0
2 0.00215         -6.50  6.37     0  2.67     0     0     0 0.563 -1.26 0.757
# i 13 more variables: r12 <dbl>, r14 <dbl>, r17 <dbl>, r18 <dbl>, r19 <dbl>,
#   r21 <dbl>, r22 <dbl>, r24 <dbl>, r28 <dbl>, r30 <dbl>, r32 <dbl>,
#   r36 <dbl>, r37 <dbl>
```

We compare below the plots AUC(lambda) given by our function and `cv.glmnet`; our manual implementation (in orange, computing the AUC as the average AUC per fold) seems to match roughly `cv.glmnet` (in green):

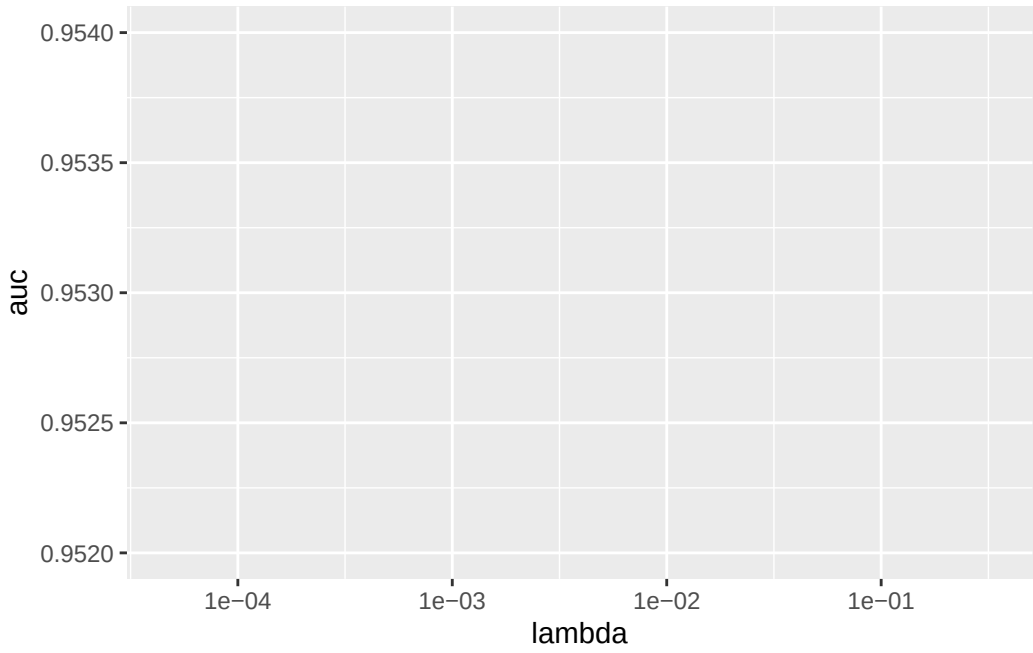
```
ggplot(res_for_plot_manual, aes(x=lambda, y = auc)) +
  geom_point(col='red') + # manual (global AUC)
  geom_point(data = res_for_plot_glmnet, aes(x=lambda, y = auc), col ='green') + # cv.glmn
  geom_point(data = res_for_plot_manual, aes(x=lambda, y = auc_indiv), col ='orange') + #
  scale_x_continuous(trans='log10') +
  coord_cartesian(ylim = c(0.89, 0.955))
```




```
ggplot(res_for_plot_manual, aes(x=lambda, y = auc)) +
  geom_point(col='red') + # manual (global AUC)
  geom_point(data = res_for_plot_glmnet, aes(x=lambda, y = auc), col ='green') + # cv.glmn
  geom_point(data = res_for_plot_manual, aes(x=lambda, y = auc_indiv), col ='orange') + #
  scale_x_continuous(trans='log10') +
  coord_cartesian(ylim = c(0.94, 0.955))
```

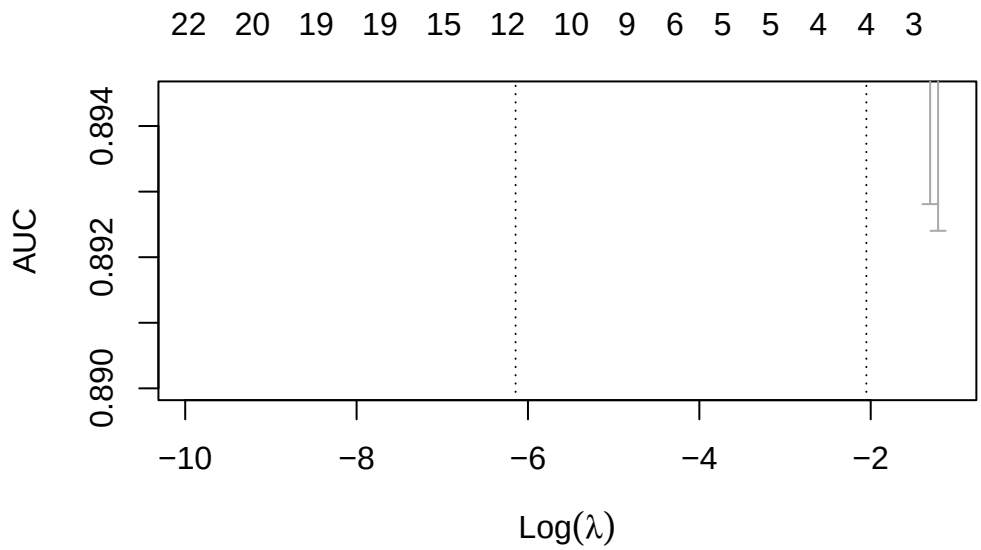



```
ggplot(res_for_plot_manual, aes(x=lambda, y = auc)) +
  geom_point(col='red') + # manual (global AUC)
  geom_point(data = res_for_plot_glmnet, aes(x=lambda, y = auc), col ='green') + # cv.glmn
  geom_point(data = res_for_plot_manual, aes(x=lambda, y = auc_indiv), col ='orange') + #
  scale_x_continuous(trans='log10') +
  coord_cartesian(ylim = c(0.952, 0.954))
```



The equivalent of 'Orange curve' (mean(fold_AUC)) using the built-in `plot.cv.glmnet`:

```
plot(default_lasso_cv, ylim = c(0.89, 0.8945))
```



10 References

Albert, A., & Aanderson, J. A. (1984). On the existence of maximum likelihood estimates in logistic regression models. *Biometrika*, *71*(1), 1–10. https://doi.org/10.1093/biomet/71.1.1

Chava, S., Alanis, E., & Shah, A. (2023). Benchmarking machine learning models to predict corporate bankruptcy. *Journal of Credit Risk*, *19*(2), 77–110. https://doi.org/10.21314/JCR.2023.002

Cornillon, P.-A., Matzner-Løber, E., Hengartner, N., & Rouvière, L. (2019). *Régression avec r - 2e édition*. EDP Sciences.

Desbois, D. (2008). Introduction to scoring methods: financial problems of farm holdings. *Case Studies in Business, Industry and Government Statistics*, *2*(1), 56–76. https://hal.science/hal-01172847

Fahrmeir, &. K., L. (1986). Asymptotic inference in discrete response models. *Statistische Hefte*, *27*(1), 179–205. https://doi.org/10.1007/bf02932567

Gourieroux, &. M., C. (1981). Asymptotic properties of the maximum likelihood estimator in dichotomous logit models. *Journal of Econometrics*, *17*(1), 83–97. https://doi.org/10.1016/0304-4076(81)90060-9

Hastie, T., Tibshirani, R., & Friedman, J. (2009). *The elements of statistical learning*. Springer New York.

Hosmer, D. W., Lemeshow, S., & Sturdivant, R. X. (2013). Applied logistic regression. In *Wiley Series in Probability and Statistics*.

James, G., Witten, D., Hastie, T., & Tibshirani, R. (2021). *An introduction to statistical learning: With applications in r*. Springer US. https://doi.org/10.1007/978-1-0716-1418

Nelder, J. A., & Wedderburn, R. W. M. (1972). Generalized linear models. *Journal of the Royal Statistical Society. Series A (General)*, *135*(3), 370. https://doi.org/10.2307/2344614

Tian, S., Yu, Y., & Guo, H. (2015). Variable selection and corporate bankruptcy forecasts. *Journal of Banking & Finance*, *52*, 89–100. https://doi.org/0.1016/j.jbankfin.2014.12.003