Linear Regression - Intuitions

CITS4009 Computational Data Analysis

Unit Coordinator: Dr Du Huynh

Department of Computer Science and Software Engineering
The University of Western Australia

Semester 2, 2022

What is linear regression?

Suppose that $y[i]$ is the numeric quantity we want to predict, and $x[i,]$ is a row of inputs that corresponds to output $y[i]$. Linear regression finds a fit function $f_\theta(x)$ such that

$$y[i] \sim f_\theta(x[i,]) = \beta_0 + \beta_1 x[i,1] + ...\beta_n x[i,n]$$

- Linear regression is the *go-to* statistical modelling method for quantities.
- You should always try linear regression first, and only use more complicated methods if they actually outperform a linear regression model.

Linear regression

$$\hat{y}[i] = f_\theta(x_i) = \beta_0 + \beta_1 x[i,1] + \beta_2 x[i,2] + ...\beta_n x[i,n] + u[i]$$

where θ contains the unknown coefficients $\beta_0, \beta_1, \beta_2, \cdots, \beta_n$ that we need to estimate. The basic assumption is that $\hat{y}$ is linear in the values of x:

- a change in the value of $x[i,m]$ by one unit (while holding all the other $x[i,k]$'s constant) will always change the value of $\hat{y}[i]$ by the amount β_m, no matter what the starting value of $x[i,m]$ was.

This is easier to see in one dimension.

- If $\hat{y} = 3 + 2x$, and if we increase x by 1, then y will always increase by 2, no matter what the starting value of x is.
- This wouldn't be true for, say, $\hat{y} = 3 + 2x^2$.

The last term, $u[i]$, represents the so-called *unsystematic errors*, or *noise*. Unsystematic errors average to 0 and are uncorrelated with $x[i,]$ and $\hat{y}[i]$.

Linear Regression - Terminologies

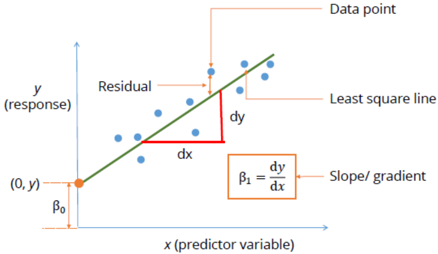
predictor, 'x-variable', independent variable, explanatory variable

coefficient

$$Y = \underbrace{\beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_p x_p}_{\text{linear predictor}} + \varepsilon$$

response, dependent variable, observation, 'y-variable'

random error, "noise"



Residual

$$\boxed{\underset{\text{Actual value}}{y} = \underbrace{\beta_0 + \beta_1 x}_{\text{Predicted value}} + \overset{\text{Residual}}{u}}$$

How can we find the best fit?

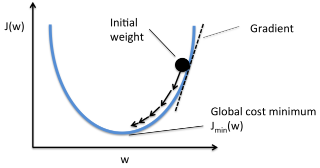
Mean Squared Error

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}[i] - y[i])^2$$

Loss Function obtained from MSE

$$J(\theta) = \frac{1}{2n}\sum_{i=1}^{n}(f_\theta(x[i]) - y[i])^2$$

(the $\frac{1}{2}$ term is introduced to cancel the factor of 2 when $J(\theta)$ is differentiated w.r.t. θ)



Building and Interpreting Linear Regression Models

CITS4009 Computational Data Analysis

Unit Coordinator: Dr Du Huynh

Department of Computer Science and Software Engineering
The University of Western Australia

Semester 2, 2022

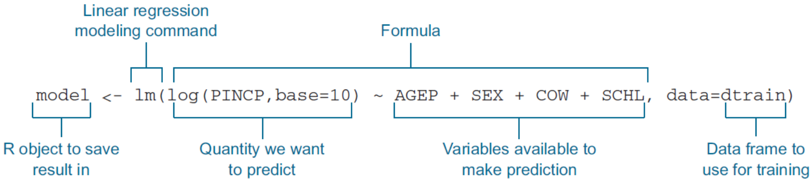
Building Linear Regression Models

Building a linear regression model

Statisticians call

- the quantity to be predicted the *dependent variable* (y) and
- the variables/columns/features used to make the predictions the *independent variables* (x's).

This can be expressed as a formula in R: $y \sim x_1 + \cdots + x_n$
For example,



Loading data and training an `lm()` model

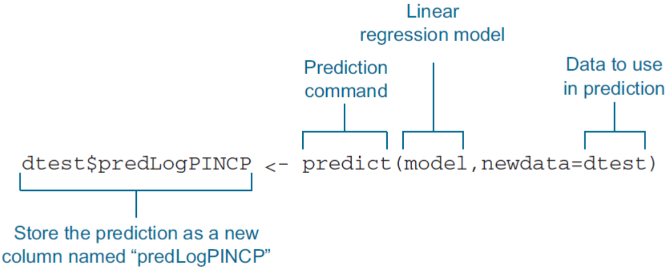
Example: Predict the log base 10 of `income` as a function of `age`, `sex`, `employment class`, and `education`.

```
# download https://github.com/WinVector/PDSwR2/raw/master/PUMS/psub.RDS
path <- "../../data_v2/PUMS/"
psub <- readRDS(paste0(path,"psub.RDS"))  # load in the data frame psub
cat("Dimension of the table frame psub =", dim(psub))
## Dimension of the table frame psub = 22241 204

set.seed(3454351)
gp <- runif(nrow(psub))
dtrain <- subset(psub, gp >= 0.5)  # split 50-50 into training
dtest <- subset(psub, gp < 0.5)    # and testing sets

# perform linear regression
model <- lm(log(PINCP,base=10) ~ AGEP + SEX + COW + SCHL, data=dtrain)
```

(`glm()` is a more generalised version of `lm()`, e.g., it allows the regression results to be mapped via a family of functions by using the keyword argument `family`)

Making predictions

```
dtrain$predLogPINCP <- predict(model, newdata=dtrain)
dtest$predLogPINCP <- predict(model, newdata=dtest)
```



The line of perfect prediction

Plotting the actual y you're trying to predict as if it were a function of your prediction.

If the predictions are very good, then the plot will have dots arranged near the line $y = x$, which we call **the line of perfect prediction**.

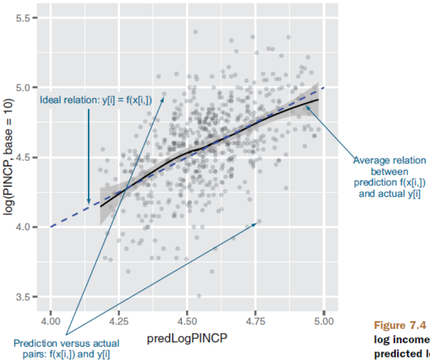
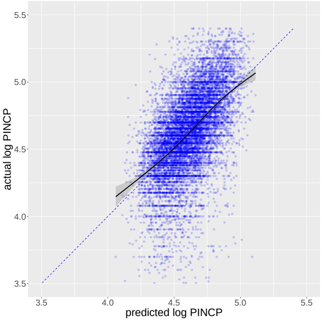


Figure 7.4 Plot of actual log income as a function of predicted log income

Characterising prediction quality

```
ggplot(data=dtest, aes(x=predLogPINCP, y=log(PINCP,base=10))) +
  geom_point(alpha=0.2, color="blue") + geom_smooth(color="black") +
  geom_line(data=dtest, aes(x=log(PINCP,base=10), y=log(PINCP,base=10)),
            color="blue", linetype=2) +
  scale_x_continuous(limits=c(3.5,5.5)) + scale_y_continuous(limits=c(3.5,5.5)) +
  labs(x="predicted log PINCP", y="actual log PINCP") + theme(text=element_text(size
```

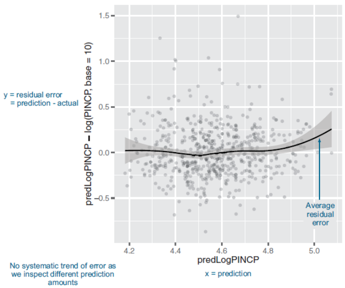


Residual Graph

A residual graph gives a sense of when the model may be underpredicting or overpredicting. A residual graph has

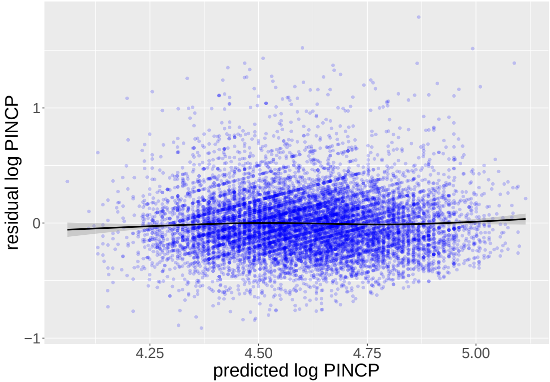
- the vertical axis being the *residual*, i.e., predictedValue − actualValue.
- the horizontal axis being the predictedValue.

The line of perfect prediction is when the residual is perfectly zero, i.e., $y = 0$.



Residuals income as a function of predicted log income

```
ggplot(data=dtest, aes(x=predLogPINCP, y=predLogPINCP-log(PINCP,base=10))) +
  geom_point(alpha=0.2, color="blue") + geom_smooth(color="black") +
  labs(x="predicted log PINCP", y="residual log PINCP") + theme(text=element_text(s
```



Interpreting Coefficients

Finding relations and extracting advice

All of the information in a linear regression model is stored in a block of numbers called the `coefficients`, available through the `coefficients(model)` command.

Apart from predicting income, we can use these coefficients to find the value of having a bachelor's degree.

```
coefficients(model)
##                        (Intercept)                                AGEP
##                         4.00588563                          0.01159846
##                          SEXFemale      COWFederal government employee
##                        -0.10768834                          0.06386719
##       COWLocal government employee COWPrivate not-for-profit employee
##                        -0.02970932                         -0.03301963
##      COWSelf employed incorporated  COWSelf employed not incorporated
##                         0.01454745                         -0.12822845
##       COWState government employee    SCHLRegular high school diploma
##                        -0.04795709                          0.11353857
##  SCHLGED or alternative credential SCHLsome college credit, no degree
##                         0.12166699                          0.18382783
##             SCHLAssociate's degree              SCHLBachelor's degree
##                         0.23870449                          0.36371138
##                SCHLMaster's degree            SCHLProfessional degree
##                         0.44457769                          0.51111666
##               SCHLDoctorate degree
##                         0.48187005
```

Interpreting the coefficients and extracting advice

The SCHL variable in the dataset has 9 levels:

```
levels(dtrain$SCHL)
## [1] "no high school diploma"         "Regular high school diploma"
## [3] "GED or alternative credential"  "some college credit, no degree"
## [5] "Associate's degree"             "Bachelor's degree"
## [7] "Master's degree"                "Professional degree"
## [9] "Doctorate degree"
```

`lm()` omits the `no high school diploma` level and converts the remaining 8 levels into variables starting with the name `SCHL`. The level that isn't shown is called the **reference level**. The coefficients of the other levels are measured with respect to the reference level. E.g., for `SCHLBachelor's degree`, the 0.3637114 coefficient means

> *"The model gives a 0.36 bonus to* $\log_{10}$ *income for having a bachelor's degree, relative to not having a high school degree."*

The income ratio between someone with a bachelor's degree and the equivalent person (same sex, age, and class of work) without a high school degree is about $10^{0.36}$ (or 2.29) times higher. So the advice is: "*college is worth it if you can find a job*".

Interpreting the coefficients and extracting advice (cont.)

Similarly, the COW and SEX variables have 7 and 2 levels respectively:

```
levels(dtrain$COW)
## [1] "Employee of a private for profit" "Federal government employee"
## [3] "Local government employee"        "Private not-for-profit employee"
## [5] "Self employed incorporated"       "Self employed not incorporated"
## [7] "State government employee"
levels(dtrain$SEX)
## [1] "Male"   "Female"
```

The first level "Employee of a private for profit" under the original COW variable becomes the reference level while the remaining 6 levels become variables measured with respect to the reference level. The first level "Male" under the SEX variable becomes the reference level, leaving SEXFemale as the variable for linear regression.

Altogether there are 16 variables (1 from AGEP, 1 from SEX, 6 from COW, and 7 from SCHL). Including the *intercept* (the β_0 coefficient), there are 17 coefficients in the model.

Model summary using summary(model)

```
summary(model)
##
## Call:
## lm(formula = log(PINCP, base = 10) ~ AGEP + SEX + COW + SCHL,
##     data = dtrain)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -1.5038 -0.1354  0.0187  0.1710  0.9741
##
## Coefficients:
##                                      Estimate Std. Error t value Pr(>|t|)
## (Intercept)                         4.0058856  0.0144265 277.676  < 2e-16 ***
## AGEP                                0.0115985  0.0003032  38.259  < 2e-16 ***
## SEXFemale                          -0.1076883  0.0052567 -20.486  < 2e-16 ***
## COWFederal government employee      0.0638672  0.0157521   4.055 5.06e-05 ***
## COWLocal government employee       -0.0297093  0.0107370  -2.767 0.005667 **
## COWPrivate not-for-profit employee -0.0330196  0.0102449  -3.223 0.001272 **
## COWSelf employed incorporated       0.0145475  0.0164742   0.883 0.377232
## COWSelf employed not incorporated  -0.1282285  0.0134708  -9.519  < 2e-16 ***
## COWState government employee       -0.0479571  0.0123275  -3.890 0.000101 ***
## SCHLRegular high school diploma     0.1135386  0.0107236  10.588  < 2e-16 ***
## SCHLGED or alternative credential   0.1216670  0.0173038   7.031 2.17e-12 ***
## SCHLsome college credit, no degree  0.1838278  0.0106461  17.267  < 2e-16 ***
## SCHLAssociate's degree              0.2387045  0.0123568  19.318  < 2e-16 ***
## SCHLBachelor's degree               0.3637114  0.0105810  34.374  < 2e-16 ***
## SCHLMaster's degree                 0.4445777  0.0127100  34.978  < 2e-16 ***
## SCHLProfessional degree             0.5111167  0.0201800  25.328  < 2e-16 ***
## SCHLDoctorate degree                0.4818700  0.0245162  19.655  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

Model summary – Explanation

The *Residuals* part of the summary show the 0th (i.e., minimum value), 25th, 50th, 75th, and 100th (i.e., maximum value) percentiles of the residuals from the model's estimations.

The *Coefficients* part of the summary is a table having 4 columns:

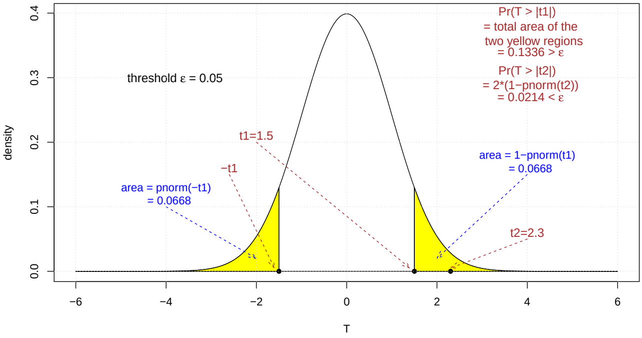
- The row names correspond to the variable names
- Column 1: estimates of the coefficients for the variables. A large positive coefficient value indicates that the variable is more significant in giving a high log income.
- Column 2: the standard error of the estimates given in column 1. A large standard error means that the corresponding estimate is not reliable (we want standard error to be small).
- Column 3: the t value in each row tells us how far the coefficient estimate is from zero (in units of likely error). It is calculated as `Estimate / Standard.Error` (we want large t value).

Model summary – Explanation (cont.)

The *Coefficients* part of the summary is a table having 4 columns: (cont.)

- Column 5: this column stores the so-called p-value, which is defined as *the probability that the estimate (in column 1) having such a t value (in column 3) being by mere chance*. In our case, we want small p-value (less than a given threshold, usually a small number such as 0.05).

Model summary – Explanation (cont.)



In the example above, we favour the estimate that gives the t value `t2` as its p-value $=$ `Pr(T > |t2|)` is smaller than the threshold ϵ. A smaller p-value indicates that the estimate is more reliable.

Model summary – explanation (cont.)

```
coef_df <- data.frame(summary(model)$coefficients)  # extract the coefficients
str(coef_df, vec.len=8)      # coef_df is a data frame having 4 columns
## 'data.frame':    17 obs. of  4 variables:
##  $ Estimate  : num  4.0059 0.0116 -0.1077 0.0639 -0.0297 -0.033 0.0145 -0.1282 -
##  $ Std..Error: num  0.014426 0.000303 0.005257 0.015752 0.010737 0.010245 0.0164
##  $ t.value   : num  277.676 38.259 -20.486 4.055 -2.767 -3.223 0.883 -9.519 -3.8
##  $ Pr...t..  : num  0.00 4.30e-301 1.36e-91 5.06e-05 5.67e-03 1.27e-03 3.77e-01

# how the t-value column is computed? We can compare the computed results below
# with the output from str()
t_value <- coef_df$Estimate / coef_df$`Std..Error`;   cat(t_value[1:8], "\n")
## 277.6764 38.25942 -20.48588 4.05451 -2.766996 -3.22304 0.8830427 -9.518971

# how the p-value column is computed?
p_value <- 2*(1 - pnorm(abs(coef_df$t.value)));   cat(p_value[1:8], "\n")
## 0 0 0 5.023949e-05 0.005657548 0.001268376 0.3772132 0
```

In the model summary, variable `COWSelf employed incorporated` has the largest *p*-value of 0.377232 (which is $> \epsilon$). The coefficient estimate 0.0145475 of this variable is considered to be *insignificant* (notice that it doesn't have any $*$ labelled next to it).

R STORES TRAINING DATA IN THE MODEL

R holds a copy of the training data in its model to supply the residual information seen in `summary(model)`.

Holding a copy of the data this way is not strictly necessary and can needlessly run you out of memory.

- You can mitigate this problem somewhat by setting all the parameters `model`, `x`, `y`, and `qr` to `FALSE` in the `lm()` call.
- If you're running low on memory (or swapping), you can dispose of R objects like model using the `rm()` command. In this case, you'd dispose of the model by running `rm("model")`.

How good is the fitted line?

To assess how good the fitted line is, we need an evaluation measure. *R-squared* (a.k.a. *coefficient of determination*) is the most commonly used measure. It can be thought of as what fraction of the y variation is explained by the model.

- Step 1: compute the total sum of squares (proportional to the variance of the data)

$$\mathrm{SS_{tot}} = \sum_{i=1}^{n} (y[i] - \bar{y})^2, \text{ where } \bar{y} = \frac{1}{n} \sum_{i=1}^{n} y[i]$$

- Step 2: compute the sum of squares of residuals

$$\mathrm{SS_{res}} = \sum_{i=1}^{n} (y[i] - \hat{y}[i])^2$$

How good is the fitted line? (cont.)

The formula for R-squared is given by:

$$R^2 \equiv 1 - \frac{\mathrm{SS}_{\mathrm{res}}}{\mathrm{SS}_{\mathrm{tot}}} = 1 - \frac{\sum_i (y[i] - \hat{y}[i])^2}{\sum_i (y[i] - \bar{y})^2}$$

A large R^2 value (close to 1) is an indication of a good fit.

Compute R^2

```
rsq <- function(y, yhat) {
    1 - sum((y-yhat)^2) / sum((y-mean(y))^2)
}
```

For the PUMS income dataset, the R^2 values were quite poor for both the training and test sets. We'd like to see R^2 values higher than this (say 0.7 to 1.0).

```
# R-squared value on the training set
rsq(log(dtrain$PINCP,base=10), predict(model, newdata=dtrain))
## [1] 0.2976165

# R-squared value on the test set
rsq(log(dtest$PINCP,base=10), predict(model, newdata=dtest))
## [1] 0.2911965
```

Linear Regression - Summary

- Linear regression has trouble with datasets that
  - have a very large number of variables, or
  - categorical variables with a very large number of levels.
- For the case where the relationship between the dependent variable and the independent variables is non-linear, we can enhance linear regression by
  - adding new variables (however, watch out for the problem mentioned above) or
  - transforming variables (e.g., the log transform of the dependent variable y, but always be wary when transforming y as it changes the error model).
- Linear regression can predict well even in the presence of correlated variables, but correlated variables lower the quality of the advice.
  - overly large coefficient magnitudes,
  - overly large standard errors,
  - wrong sign on a coefficient could be indicators of correlated inputs.
- Always rechecking your model on test data.

References

Practical Data Science with R (Second Edition). *Nina Zumel and John Mount*, Manning, 2020: Chapter 7, Section 7.1 (pages 215-225); Section 7.1.4-7.1.5 (pages 228-233)

Logistic Regression

CITS4009 Computational Data Analysis

Unit Coordinator: Dr Du Huynh

Department of Computer Science and Software Engineering
The University of Western Australia

Semester 2, 2022

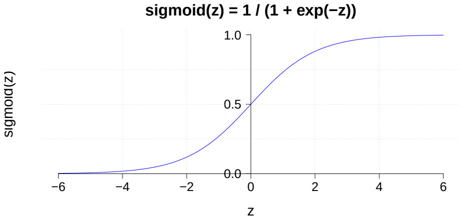
What is logistic regression?

Logistic regression is linear regression with a *sigmoid* function for mapping the regression values to the $[0, 1]$ interval – i.e., probabilities. Because of that, logistic regression is commonly used for binary classification. It is the most popular and simple machine learning classification technique.

In logistic regression classification, given a row of inputs $x[i,]$ belonging to a given class C, we find $f_\theta(x)$ such that

$$P(x[i,] \in \text{class } C) \sim \sigma(f_\theta(x[i,])) = \sigma(\beta_0 + \beta_1 x[i,1] + \cdots + \beta_n x[i,n])$$

where $\sigma(z) = \frac{1}{1+e^{-z}}$ is known as the *sigmoid* function:



The inverse of the sigmoid is the `logit`

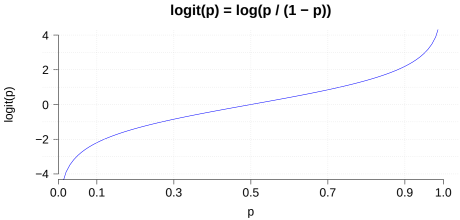
Unlike the linear regression example earlier, in logistic regression classification we don't have the $f_\theta(x[i])$ value explicitly for each observed data $x[i]$; instead, we have the class label for each $x[i]$ from the training set. e.g., $p[i] \equiv \Pr(x[i] \in \text{class "flight delayed"}) = 1$ if the class label for $x[i]$ is `"flight delayed"` and $p[i] = 0$ otherwise.

In the training process, the logit function, which is the inverse of the sigmoid function is used. The `logit` function is defined as

$$\text{logit}(p) \equiv \log\left(\frac{p}{1-p}\right)$$

where p is a probability. The ratio $\frac{p}{1-p}$ is known as the *odds*.

The inverse of the sigmoid is the `logit` (cont.)



As $p \to 1$, $\text{logit}(p) \to \infty$.
As $p \to 0$, $\text{logit}(p) \to -\infty$.

- In the *flight* example, the logit is *the log of the odds* (or *log-odds*) that a flight will be delayed.
- Logistic regression assumes that `logit(p[i])` is linear in the values of `x[i]`. i.e.,
  $\text{logit}(\Pr(x[i] \in \text{class } C)) \equiv f_\theta(x[i]) = \beta_0 + \beta_1 x[i, 1] + \cdots + \beta_n x[i, n]$

Logistic regression classification

- Logistic regression is a linear regression that finds the log-odds of the probability that you're interested in.
- In Machine Learning, symbols $x[i]$ and $y[i]$ are commonly used to denote the ith input feature and the ith output that we try to predict, regardless of whether it is a regression problem or classification problem. Thus, we won't see the symbol $p[i]$ used in logistic regression classification. Instead, we can think of $\text{logit}(y[i])$ is used in the linear regression on the training set. In the prediction stage, the sigmoid function is used to convert the regression values back to probability values, which are thresholded to yield the class labels.

Example: Determining if new born babies are at risk

Example: The goal here is to identify ahead of time the risk with a high probability, so that resources can be allocated appropriately.

We use the *CDC 2010 natality* data file: http://mng.bz/pnGy

```
path <- "../../data_v2/CDC/"
load(paste0(path, "NatalRiskData.rData"))
train <- sdata[sdata$ORIGRANDGROUP <= 5,]
test <- sdata[sdata$ORIGRANDGROUP > 5,]
cat("dim(train) =", dim(train), "; dim(test) =", dim(test))
```

```
## dim(train) = 14212 15 ; dim(test) = 12101 15
```

New born babies are assessed at one minute and five minutes after birth using what's called the *Apgar test*, which is designed to determine if a baby needs immediate emergency care or extra medical attention.

- A baby who scores below 7 (on a scale from 0 to 10) on the Apgar scale needs extra attention.

Data Dictionary

Variable	Type	Description
`atRisk`	Logical	`TRUE` if 5-minute Apgar score < 7; `FALSE` otherwise
`PWGT`	Numeric	Mother's prepregnancy weight
`UPREVIS`	Numeric (integer)	Number of prenatal medical visits
`CIG_REC`	Logical	`TRUE` if smoker; `FALSE` otherwise
`GESTREC3`	Categorical	Two categories: `<37 weeks` (premature) and `>=37 weeks`

Variable	Type	Description
`DPLURAL`	Categorical	Birth plurality, three categories: `single/twin/triplet+`
`ULD_MECO`	Logical	`TRUE` if moderate/heavy fecal staining of amniotic fluid
`ULD_PRECIP`	Logical	`TRUE` for unusually short labor (< three hours)
`ULD_BREECH`	Logical	`TRUE` for breech (pelvis first) birth position
`URF_DIAB`	Logical	`TRUE` if mother is diabetic
`URF_CHYPER`	Logical	`TRUE` if mother has chronic hypertension
`URF_PHYPER`	Logical	`TRUE` if mother has pregnancy-related hypertension
`URF_ECLAM`	Logical	`TRUE` if mother experienced eclampsia: pregnancy-related seizures

Building the formula

```
complications <- c("ULD_MECO", "ULD_PRECIP", "ULD_BREECH")
riskfactors <- c("URF_DIAB", "URF_CHYPER", "URF_PHYPER", "URF_ECLAM")
y <- "atRisk"
x <- c("PWGT", "UPREVIS", "CIG_REC", "GESTREC3", "DPLURAL",
       complications, riskfactors)
fmla <- paste(y, paste(x, collapse=" + "), sep=" ~ ")
cat(fmla)
```

```
## atRisk ~ PWGT + UPREVIS + CIG_REC + GESTREC3 + DPLURAL + ULD_MECO + ULD_PRECIP +
```

Fitting the logistic regression model

```
model <- glm(fmla, data=train, family=binomial(link="logit"))
```

The `family` function specifies the assumed distribution of the dependent variable `y`.

二项分布

- In this case, `y` is modelled as a binomial distribution, or as the event of tossing a coin whose probability of showing the head depends on `x`.

线性模型

- The `link` function "links" the output to a linear model – it's as if you pass `y` through the `link` function, and then model the resulting value as a linear function of the `x` values.

Different combinations of `family` functions and `link` functions lead to different kinds of generalised linear models (e.g., `Poisson`, or `probit`).

DON'T FORGET THE FAMILY ARGUMENT!

Without an explicit `family` argument, `glm` defaults to standard linear regression (like `lm`).

Make predictions

```
train$pred <- predict(model, newdata=train, type="response")
test$pred <- predict(model, newdata=test, type="response")
```

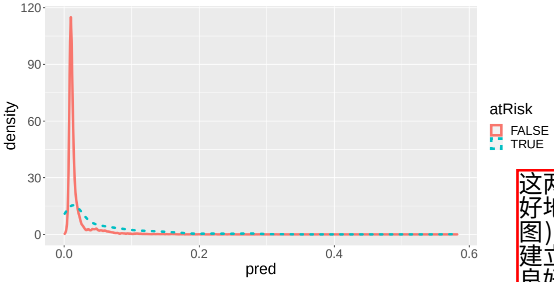
Note the additional parameter `type="response"`.

- This tells the `predict()` function to return the predicted probabilities y. 有response才返回预测概率
- If you don't specify `type="response"`, then by default `predict()` will return the output of the link function (i.e., the `logit` value).

否则返回link函数的输出（如logit

Characterising the Prediction Quality

```
library(ggplot2)
ggplot(train, aes(x=pred, color=atRisk, linetype=atRisk)) + geom_density(size=1.5) +
    theme(text=element_text(size=20))
```



这两个分布没有很好地分离(如上图)，表明模型不能建立一个同时获得良好召回率和良好精度的分类器

The two distributions not being well separated (such as the above diagram) indicates that the model cannot build a classifier that simultaneously achieves good recall and good precision.

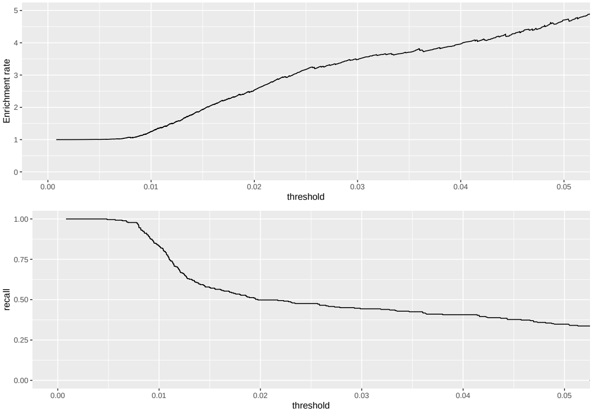
Picking the threshold for classification

```r
library(ROCR)
library(grid)
library(gridExtra)
perf <- prediction(train$pred, train$atRisk)
precObj <- performance(perf, measure="prec")
recObj <- performance(perf, measure="rec")

thresh <- (precObj@x.values)[[1]]      # threshold
precision <- (precObj@y.values)[[1]]  # precision
recall <- (recObj@y.values)[[1]]      # recall
ROCdf <- data.frame(threshold=thresh, precision=precision, recall=recall)

# Null probability
pnull <- mean(as.numeric(train$atRisk))
cat('pnull =', pnull)
## pnull = 0.01920912
```

We will plot the *enrichment rate* (defined as *the ratio of precision to the average rate of positives*) and recall versus the threshold value.

Building the plots and exploring modelling trade-offs

```
p1 <- ggplot(ROCdf, aes(x=threshold)) + geom_line(aes(y=precision/pnull)) +
  coord_cartesian(xlim = c(0,0.05), ylim=c(0,5) ) + labs(y="Enrichment rate")
p2 <- ggplot(ROCdf, aes(x=threshold)) + geom_line(aes(y=recall)) +
  coord_cartesian(xlim = c(0,0.05))
grid.arrange(p1, p2, nrow = 2)
```



Evaluating the chosen threshold and interpreting the classification quality

```
cat("Confusion matrix of 'at risk' predictions:\n")
## Confusion matrix of 'at risk' predictions:
(ctab.test <- table(actual=test$atRisk, predicted=test$pred>0.02))
##        predicted
## actual  FALSE TRUE
##   FALSE  9487 2405
##   TRUE     93  116
(precision <- ctab.test[2,2] / sum(ctab.test[,2]))   # TP / (TP+FP)
## [1] 0.04601349
(recall <- ctab.test[2,2] / sum(ctab.test[2,]))      # TP / (TP+FN)
## [1] 0.5550239
(enrich <- precision / mean(as.numeric(test$atRisk)))
## [1] 2.664159
```

- The resulting classifier has low precision.
- But it identifies a set of potential at-risk cases that contains 55.5% of the true positive cases in the test set, at a rate 2.66 times higher than the overall average.

Interpreting the coefficients

As with linear regression, every categorical variable is expanded to a set of indicator variables.

- If the original variable has n levels, there will be $n - 1$ indicator variables; the remaining level is the reference level.

Given that the coefficient for `GESTREC3 < 37` weeks (for a premature baby) is 1.545183, we can say

- For a premature baby, the odds of being at risk are $e^{1.545183} = 4.68883$ times higher compared to a baby that's born full-term, with all other input variables unchanged.

Model Summary

```
summary(model)
##
## Call:
## glm(formula = fmla, family = binomial(link = "logit"), data = train)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -0.9732  -0.1818  -0.1511  -0.1358   3.2641
##
## Coefficients:
##                          Estimate Std. Error z value Pr(>|z|)
## (Intercept)             -4.412189   0.289352 -15.249  < 2e-16 ***
## PWGT                     0.003762   0.001487   2.530 0.011417 *
## UPREVIS                 -0.063289   0.015252  -4.150 3.33e-05 ***
## CIG_RECTRUE              0.313169   0.187230   1.673 0.094398 .
## GESTREC3< 37 weeks       1.545183   0.140795  10.975  < 2e-16 ***
## DPLURALtriplet or higher 1.394193   0.498866   2.795 0.005194 **
## DPLURALtwin              0.312319   0.241088   1.295 0.195163
## ULD_MECOTRUE             0.818426   0.235798   3.471 0.000519 ***
## ULD_PRECIPTRUE           0.191720   0.357680   0.536 0.591951
## ULD_BREECHTRUE           0.749237   0.178129   4.206 2.60e-05 ***
## URF_DIABTRUE            -0.346467   0.287514  -1.205 0.228187
## URF_CHYPERTRUE           0.560025   0.389678   1.437 0.150676
## URF_PHYPERTRUE           0.161599   0.250003   0.646 0.518029
## URF_ECLAMTRUE            0.498064   0.776948   0.641 0.521489
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 2698.7  on 14211  degrees of freedom
## Residual deviance: 2463.0  on 14198  degrees of freedom
```

Null Deviance

Deviance is a measure for showing how well the model fits the data.

- It is 2 times the negative log likelihood of the dataset, given the model.

```
loglikelihood <- function(y, py) {
  sum(y * log(py) + (1-y)*log(1 - py))
}

# Null probability
(pnull <- mean(as.numeric(train$atRisk)))
## [1] 0.01920912

# Normalised Null deviance
null.dev <- -2*loglikelihood(as.numeric(train$atRisk), pnull) / nrow(train)
cat("Normalised deviance of the Null model is:", null.dev)
## Normalised deviance of the Null model is: 0.18989
```

Residual Deviance

```
# on the training set
pred <- predict(model, newdata=train, type="response")

# deviance of the logistic regression model
resid.dev <- -2*loglikelihood(as.numeric(train$atRisk), pred) / nrow(train)
cat("Normalised deviance of the logistic regression model on the training set is:\n
    resid.dev)
## Normalised deviance of the logistic regression model on the training set is:
##  0.1733037

# on the test set
pred <- predict(model, newdata=test, type="response")

# deviance of the logistic regression model
resid.dev <- -2*loglikelihood(as.numeric(test$atRisk), pred) / nrow(test)
cat("Normalised deviance of the logistic regression model on the test set is:\n",
    resid.dev)
## Normalised deviance of the logistic regression model on the test set is:
##  0.1609036
```

Akaike information criterion - AIC

The AIC is the log likelihood penalised for the model complexity (i.e., the number of coefficients). Recall that \$AIC $= \text{deviance} + 2 \times \text{numberOfParameters}$.

```r
aic <- resid.dev + 2*(length(model$coefficients))

cat("AIC value of the logistic regression model on the test set is:", aic)
## AIC value of the logistic regression model on the test set is: 28.1609
```

- The AIC is usually used to decide which and how many input variables to use in the model.
- If you train many different models with different sets of variables on the same training set, you can consider the model with the lowest AIC to be the best fit.

Logistic regression summary

- Logistic regression is the *go-to* statistical modelling method for binary classification.
- Try logistic regression first, and then more complicated methods if logistic regression doesn't perform well.
- Logistic regression is well calibrated: it reproduces the marginal probabilities of the data.
- Similar to linear regression, logistic regression
- will have trouble with datasets that have a very large number of variables or categorical variables with a very large number of levels;
- can predict well even in the presence of correlated variables, but correlated variables lower the quality of the advice.
- `glm()` provides good diagnostics, but rechecking your model on test data is still your most effective diagnostic.

Take home messages

- Build Regression models (`lm()` and `glm()`)
- Interpret coefficients
- Evaluate performance using R-Squared for linear regression
- Evaluate performance using Deviance and AIC for logistic regression

References

- **Practical Data Science with R** (Second Edition). *Nina Zumel and John Mount*, Manning, 2020: Chapter 7, Section 7.2, pages 237-256.
- Layman interpretation of R-squared and Correlation Coefficient: https://www.quora.com/Correlation-coefficient-vs-coefficient-of-determination-whats-the-differenc