Making Sense out of Software Engineering Data

Sandro Morasca

Università degli Studi dell'Insubria
Dipartmento di Scienze Teoriche e Applicate

sandro.morasca@uninsubria.it

July 10, 2018

Software Engineering strives to be like any other engineering discipline

Goals of (software) production

- high quality product
- within budget constraints
- by a specified deadline

These goals have been achieved in other production processes by using scientific principles

- hypothesis setting (based on observation)
- hypothesis verification (based on empirical studies)

Why Empirical Studies in Software Engineering?

So far, software development improvement has been carried out on a mostly ideological basis, e.g.,

- goto's considered harmful (is that true? harmful for what? to what extent?)
- object-based approaches: modules should have
- high internal cohesion (whatever that means)
  - low external coupling (whatever that means)
  - object-oriented approaches
- multiple inheritance (how many levels? how "multiple?")
- single inheritance (how many levels?)
- (no inheritance?)

They can be useful for setting hypotheses

- not as unproven assumptions

Because . . .

Software Engineering needs empirical investigations to

- substantiate claims—like in any scientific discipline
- enable continuous, quantifiable improvement—like in any engineering discipline

Software Engineering is a human-intensive business

- rigor and precision are indispensable, even more than in other disciplines . . .
- . . . but common sense should always rule

Don't Use "Opaque" Resources

Gurus

- you do not know how well-grounded their knowledge is

Proprietary methods

- you do not know what is behind the scenes

Aim of the Game of Using Quantitative Approaches in Software Engineering

Quantitative approaches can help

- plan
- predict
- monitor
- control
- evaluate

products and processes, to

- choose products and processes
- improve products and processes

Final goals

- cost reduction
- meeting deadlines
- product quality improvement

Do Start with Goals

Corporate goals

- industrial goals: e.g., reduce maintenance costs by X
- research goals: e.g., study the effect of size on effort

Tactical goals

- industrial goals: e.g., improve the software design phase
- research goals: e.g., predict development effort based on size in company Y

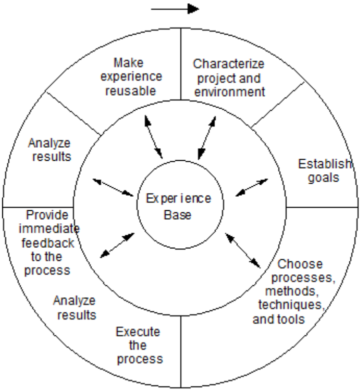
Measurement goals

Goals may somewhat change along the way

Do Have an Organized Measurement Process

Planning and quantitative analysis of the software development process

6 iteration steps: experience is adapted and reused at each iteration



Do Have an Organized Process

You are going to spend time and effort in the empirical study

Check

- resources
- availability
- timeliness
- costs
- ...

Measurement Goals

Characterizing, evaluating, estimating, predicting

- effort of a project
- time of a project
- number of faults in a software component/product
- probability of having at least one fault in a software component/product
- time to next failure
- impact of introducing a new technique
- ...

How? When? Where? Who? (Why?)

Do Have a (GQM) Goal Template

Object of Study: entity or the set of entities to study

- e.g., a software specification, or a testing process

Purpose: reason/type of result that should be obtained

- e.g., characterization, evaluation, prediction, improvement

Quality Focus: attribute or set of attributes to study

- e.g., size (for the software specification), or effectiveness (for the testing process)

Point of View: person or organization for whose benefit measurement is carried out

- e.g., the designers (for the software specification), or the testers (for the testing process)

Environment: the context (e.g., the specific project or environment) in which measurement is carried out

Do Say What You Mean and Mean What You Say

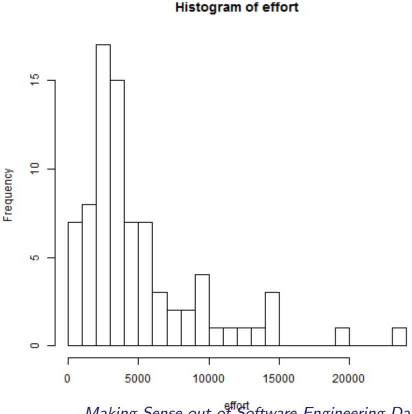
- Object of study: high-level design of software systems
- Purpose: prediction
- Quality focus: maintainability
- Viewpoint: project leader and development team
- Environment: agile development at site X of company Y

The dimensions of a GQM goal are not just words

- they are the specification of your entire study
- each of them influences what you will do

A distribution, descriptive statistics

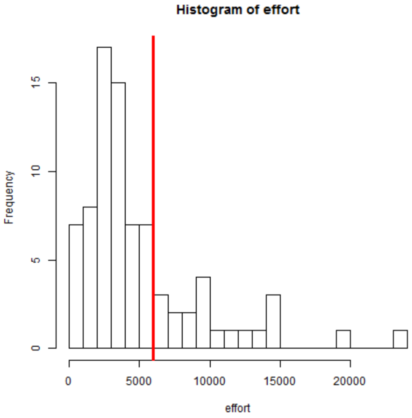
	n	min	max	med	m	σ
Effort	81	546	23940	3647	5046.31	4419.77



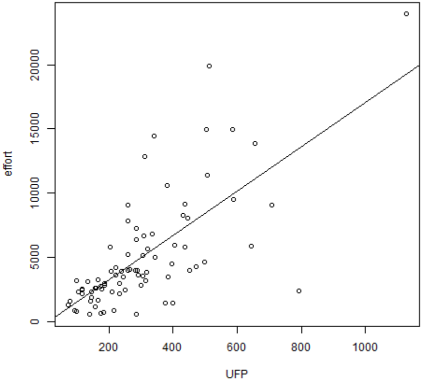
Evaluation and Statistics

A distribution, descriptive statistics, an evaluation criterion

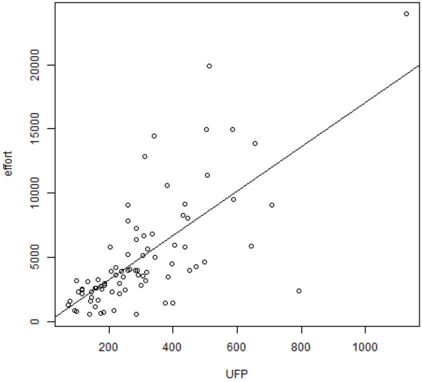
- below 6000: good
- above 6000: bad



A statistical association



A **causal** statistical association



Do Make Empirical Hypotheses Explicit

Prediction and improvement purposes require empirical hypotheses, e.g.,

- the higher the size, the higher the effort
- the higher the size, the higher the fault-proneness
- the higher the class cohesion, the higher the class maintainability
- the higher the class coupling, the lower the class maintainability

Now, how do we measure

- effort? cohesion? coupling?
- fault-proneness? maintainability?

First Question

Are we measuring the right attribute?

- (usefulness)

Many attributes (qualities) are used to speak about software artifacts

- size
- complexity
- cohesion, coupling, connectivity
- functionality
- maintainability, reliability, usability . . .

Many techniques are defined to improve software with respect to software attributes, e.g.,

- decrease coupling/increase cohesion to increase maintainability

Second Question

Are we measuring the attribute right?

- (construct validity)

Thousands of measures have been defined for software attributes

However, we need a clear idea of what measures for an attribute should look like when defining a measure for that software attribute

Acceptance of a measure should not be based on a matter of belief, a leap of faith

Internal Software Attributes

Those attributes of a software artifact

- *that can be measured based only on the knowledge of the artifact*

Examples: size, structural complexity, coupling, cohesion
Conventional wisdom has it that they are

- easy to measure
- formally characterized
  - Measurement Theory, Axiomatic Approaches
- almost useless *per se*: they need to be linked to some
  - external software attribute, or
  - process attribute

External Software Attributes

Those attributes of an artifact

- *that cannot be measured based only on the knowledge of the artifact*

They can be measured based on the knowledge of

- the artifact
- its "environment"
- the interactions between the artifact and the environment

Examples: reliability, usability, maintainability, portability

External Software Attributes

Conventional wisdom has it that they are

- hard to measure
- not formally characterized
- useful *per se*
    - relevant for some kind of "user" of the artifact
        - end users, developers, computers

Measure

A measure m for an attribute associates a value with each entity

- $m : E \to V$

where

- E: set of entities
- V: set of values

Depending on the measure, the set of values may be

- numeric (continuous or discrete) or
- non-numeric

Don't: Never Mind the Measures—What about the Numbers!

Do not use a measure just because it is there

- it could be a waste of time and effort
- it could be misleading

Do not use a measure just because everybody is using it

- we need to get rid of old, ineffective measures

A measure is just a function

- a number-producing machine

Do Theoretically Validate the Measures

Make sure your measures make sense before using them

Measurement Theory

- general framework

Axiomatic Approaches

- Weyuker's
  - Complexity
- Briand, Morasca, and Basili's
  - Size
  - Structural Complexity
  - Cohesion
  - Coupling

Scale

Technically, a scale is a triple

- an Empirical Relational System
- a Numerical Relational System
- a measure that satisfies the Representation Condition

In other words

- a scale is a measure that makes sense
- e.g., Size(A) $>$ Size(B) if and only if A is intuitively longer than B

Don't Engage in Nitpicking

Is Lines Of Code (LOC) a scale?

Fragment A:

```
i++; j++; h++;
```

Fragment B:

```
i++;
j++;
```

Representation Condition

- $A \ LONGER\_THAN \ B \Leftrightarrow LOC(A) > LOC(B)$

FALSE!

Don't: If It Is not Size, It Must Be Complexity

Do not use

- blanket terms
- different terms for the same concepts

Complexity has often been used as an umbrella term for

- coupling
- lack of cohesion
- connectivity
- information
- size

Don't Unsuccessfully Validate Your Measures

Weyuker's axioms were defined for

- software bodies, the executable parts of software programs
- with concatenation as the only possible operation
- for software complexity

Why validate a class coupling measure with Weyuker's axioms?

Take Briand, Morasca, and Basili's coupling axioms

Suppose that the class coupling measure satisfies all of them, except one

Why say that the class coupling measure was validated with Briand, Morasca, and Basili's coupling axioms?

Don't Try to Impress

Do not introduce esoteric attributes

- they may be second-order ones
- size is typically the most important attribute
- they may already exist under different names

Do not introduce complicated measures for (esoteric) attributes

- they could be very difficult to collect
- they may be highly correlated with existing, easier to quantify measures
- they may be ineffective

Don't Discard Subjective Scales

Objective scale

- Lines of Code

Subjective scale

- an instructor's grading of programs

Conventional wisdom: "objective scales are always better than subjective scales"

Well ...

- subjective scales may provide important information
- assessment is always subjective
- decisions are always subjective

(Don't) Use Surrogate Measures

Sometimes, we do not really have the measure we want, so we come up with a surrogate measure

- number of faults instead of cost required to fix them
- LOC instead of maintainability cost
- LOC instead of maintainability itself
- estimate of the number of faults instead of the number of faults
- LOC instead of FP

Buyer beware!

Do Use Models as External Measures

According to Measurement Theory, the right way of quantifying an external attribute is to use a probability model

Think reliability: probability of

- conditioned event: occurrence of no failures until time t
- conditioning event: a given software, a given way of using it

Think maintainability: probability of

- conditioned event: maintenance completed by time t
- conditioning event: a given software, a given way of maintaining it, a given maintenance requirement

There may be different models (i.e., measures) for the same external attribute-like with internal attributes

Don't Use Other Measures instead of Models

Reliability is not the average time between failures

- that is derived from the reliability model

Maintainability is not the cost of maintaining a program

GQM Goal: Effort

Object of study: software modules
Purpose: prediction
Quality focus: effort
Viewpoint: project leader and development team
Environment: development site X of company Y

We use the desharnais1_1_1 dataset from the PROMISE repository

GQM Goal: Fault-proneness

Object of study: software modules
Purpose: prediction
Quality focus: fault-proneness
Viewpoint: project leader and development team
Environment: development site X of company Y

We use the mc2 dataset from the PROMISE repository

Data Analysis

When life gives you lemons, make lemonade

- you may not get all the data you would like
- you may not get all the quality data you would like

Do not strive for perfection

- It may be too expensive to get the quality data you would like

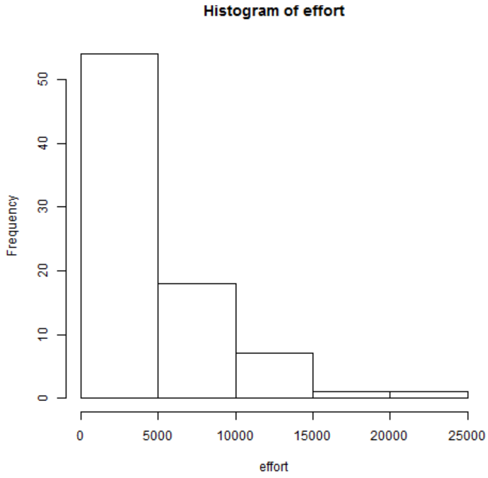
Do Take a Look at the Data

Before doing any analysis, take a look at the data to see what they look like

R: Get the Data

```
#Upload a dataset
desharnais <-
    read.csv(file="j:\\RData\\desharnaisnew.txt",
    head=FALSE,sep=",")

#Look at the dataset
desharnais

#Select the effort column
desharnais[6]
#Show the histogram of the effort column: Error!
hist(desharnais[6])
#Extract a column vector from the data frame
desharnais[[6]]

#Assign the column vector to a variable
effort <- desharnais[[6]]
```

Simple Histogram



R: Histograms


```
#Show histograms of effort with different numbers of
    bars
hist(effort)
hist(effort, breaks = 10)
hist(effort, breaks = 20)
```

R: Operations on Vectors

```
#Show one element of the vector
effort[3]

#Show a segment of the vector
effort[3:5]

#Show specific elements of the vector: Error!
effort[3,5,7]

#Build a sequence of values
projects <- c(3,5,7)

#Use the sequence of values as indices to select
    specific elements of the vector
effort[projects]
```

R: Operations on Vectors

```
#Sum
totalEffort <- sum(effort)
totalEffort

#Mean
meanEffort <- mean(effort)
meanEffort

#Product by a scalar
monthlyCostA <- 1000
costA <- monthlyCostA*effort
costA

monthlyCostB <- 200
costB <- monthlyCostB*effort
costB
```

R: Operations on Vectors


```
#Vector difference
difference <- costA - costB
difference

#Concatenation of vectors
small <- effort[1:6]
small
vec <- c(small, 4444)
vec
smaller <- effort[11:13]
vec <- c(small, smaller)
vec
```

R: Operations on Vectors


```
#Removal of elements
newvec <- vec[-3]
newvec

#Removal of elements
newvec <- vec[-c(1,3,4)]
newvec

#Select a matrix
mat <- desharnais[3:5]
mat

#Selections within the matrix
mat[80,3]
mat[80,]
mat[,3]
```

R: Operations on Vectors


```
#Logical operations on vectors
effort > meanEffort
effort[effort > meanEffort]
effort[effort > meanEffort | effort < 1000]
```

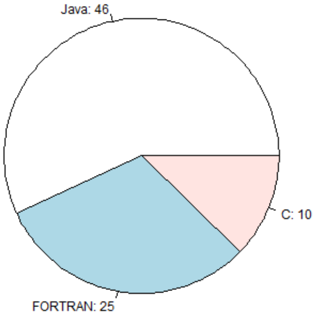
R: Array

```
#Build an array
dummyLanguage <- array( 1, dim = 46 )
dummyLanguage

otherDummyLanguage <- array(2, dim = 25)
otherDummyLanguage

#Concatenate
dummyLanguage <- c(dummyLanguage, otherDummyLanguage)
dummyLanguage

dummyLanguage <- c(dummyLanguage, array( 3, dim = 10 ))
dummyLanguage

dummyLanguage <- c(dummyLanguage, array( 4, dim = 19 ))
dummyLanguage
```

R: Matrix

```
#Build a matrix
mat <- matrix(nrow = 2, ncol = 2)
mat[1,1] <- 231
mat[1,2] <- 20
mat[2,1] <- 85
mat[2,2] <- 16
mat

mat[1]
mat[2]
mat[3]
mat[4]
```

Pie Chart



R: Basic Plots


```
#Select the language column
language <- desharnais[[12]]

#Use pie chart. Something is not quite right ...
pie(language)

#Summarize variable
frequencies <- table(language)

frequencies

#Extract the names of the rows of a table
row.names(frequencies)

#Pie chart with basic labels
pie(frequencies)
```

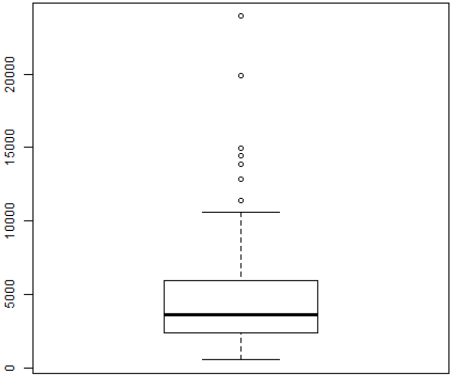
R: Some String Manipulation


```
#String manipulation
firstName <- "Sandro"
familyName <- "Morasca"
completeName <- paste(firstName, familyName, sep = " ")
completeName
```

R: Basic Plots


```
#Pie chart with labels
lbls <- paste(row.names(frequencies), frequencies, sep
    = ": ")
pie(frequencies, labels = lbls)

#Change names of rows
newNames <- c("Java", "FORTRAN", "C")
row.names(frequencies) <- newNames
frequencies

#Pie chart with right labels
lbls <- paste(row.names(frequencies), frequencies, sep
    = ": ")
pie(frequencies, labels = lbls)
```

Single Boxplot



Do Provide Descriptive Statistics and Values

Check and provide descriptive statistics for at least

- cardinality of the data set
- indicator of central tendency
- indicator of dispersion

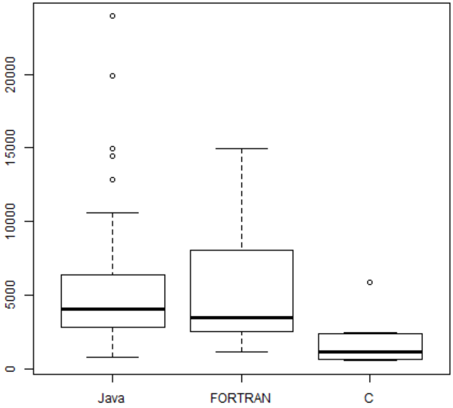
	n	min	max	med	m	σ
Effort	81	546	23940	3647	5046.31	4419.77

The ones to use will depend on the specific type of scale

Figures often provide only an intuitive idea of the results

- add a table to provide the actual values
- explain them in the text

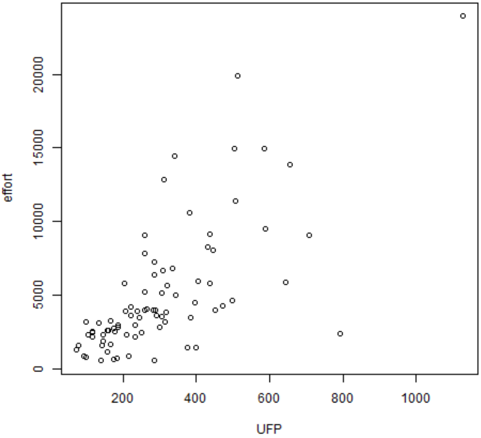
Multiple Boxplots



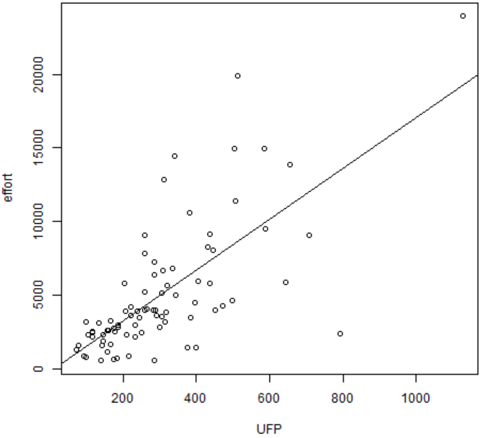
R: Basic Plots


```
#Single boxplot
boxplot(effort)

#Multiple boxplot
boxplot(V6~V12,desharnais)
#Alternatively
boxplot(effort~language)

#Better
boxplot(effort~language, names = newNames)

#Rotations
boxplot(effort~language,las = 0, names = newNames)
boxplot(effort~language,las = 1, names = newNames)
boxplot(effort~language,las = 2, names = newNames)
boxplot(effort~language,las = 3, names = newNames)
```

Scatterplot



Scatterplot with Regression Line



R: Basic Plots

```
#Select Unadjusted Function Points vector
UFP <- desharnais[[9]]

#Building a scatterplot
plot(UFP, effort)

#Building a linear model
fit <- lm(effort~UFP)

#Adding the regression line to the scatterplot
abline(fit)
```

R: Basic Plots

```
#Save a diagram onto a file
png("c:\\RDiagrams\\scatterplot.png")

#Building a scatterplot
plot(UFP, effort)

#Building a linear model
fit <- lm(effort~UFP)

#Adding the regression line to the scatterplot
abline(fit)

dev.off()
```

Levels of Measurement/Scale Types

Five scale types/levels of measurement are usually identified

- nominal
  - least informative one
- largest set of admissible transformations
- ordinal
- interval
- ratio
- absolute
  - most informative one
  - smallest set of admissible transformations

Nominal Scales

Examples:

- (non-Computer-Science): gender, species, type of vehicle
- (Computer Science): programming language, CASE tool, development process

Values: labels

Use: classification

Invariance: set of equivalence classes identified by the labels

- that is a first piece of information
- we can use whatever labels we wish, even numbers, but
- the order among them has no meaning
- arithmetic operations on them have no meaning

Nominal Scales Descriptive Statistics

Example: a software system is composed of modules written in four languages:

- Java (46%), FORTRAN(25%), C (10%), Ada (19%)

Cardinality of the data set: 100 modules

Frequencies: p(v)

- p(Java) = .46
- p(FORTRAN) = .25
- p(C) = .10
- p(Ada) = .19

R: Operations on Nominal Variables

```
#Compute the frequencies
frequencies <- table(dummyLanguage)
newNames <- c("Java", "FORTRAN", "C", "Ada")
row.names(frequencies) <- newNames
frequencies
```

Nominal Scales Central Tendency Statistics

Mode: one of the most likely values (there may be more than one mode)

- $mode \in V(\forall v \in V \;\; p(mode) \geq p(v))$

It is the value on which it makes the most sense to bet on when we select one value, if no additional information is available

- if we have to bet on the language used to write a module picked at random, we would choose Java

R: Operations on Nominal Variables

```
#Compute the mode: index of the language associated
    with the maximum frequency
maxIndex <- which.max(frequencies)
maxIndex
#Retrieve the name of the language associated with the
    maximum frequency
maxLanguage <- names(maxIndex)
maxLanguage
```

R: Writing a Function


```
#Build a function
mode <- function(x)
{
  frequencies <- table(x)
  return (names(which.max(frequencies)))
}

#Call a function
mode(dummyLanguage)
```

Nominal Scales Dispersion Statistics

Information content

- it measures the
  - degree of uncertainty/lack of knowledge associated with a probability distribution
  - amount of information provided by a random experiment about a probability distribution

Definition formula

- $H(p) = -\sum_{v \in V} p(v)\ log_2 p(v)$

Nominal Scales Dispersion Statistics

H(p) was derived based on axioms, including

- it is minimum ($H(p) = 0$) when there is perfect certainty on the outcome of the random experiment, i.e.,
- $p(v) = 1$ for some $v \in V$ and $p(u) = 0$ for any other uV
- it is maximum ($H(p) = log_2 n$) when all of the n values are equally likely
- ...

Example: $H(p) = 1.802755$

R: Writing a Long Function

```
probs <- frequencies/sum(frequencies)

#Long version of function that computes entropy
infoLong <- function(p)
{ info <- 0
  for(i in 1:length(p)){
    if(p[[i]] != 0){ entropy <- -p[[i]] * log2(p[[i]])
    }else{ entropy <- 0 }
    info <- info + entropy
  }
  return (info)
}

infoLong(probs)
#Just checking ...
-(0.46*log2(0.46)+0.25*log2(0.25)+0.1*log2(0.1)+0.19*log2(
```

R: Writing a Compact Function


```
#Short, more complete version of function that
    computes entropy
info <- function(p)
{
  p.norm <- p[p>0]/sum(p)
  return (-sum(log2(p.norm)*p.norm))
}

info(probs)
info(frequencies)
```

Nominal Scales Dispersion Statistics

Gini's impurity index is a measure of dispersion

- $G(p) = 1 - \sum_{v \in V} p^2(v)$

It measures how often a randomly chosen element from a set would be incorrectly categorized if it were randomly categorized according to the distribution of categories in the set

- $G(p) = \sum_{v \in V} p(v)(1 - p(v))$

Example: $G(p) = 0.6798$

R: A Function for Gini


```
#Function that computes Gini
gini <- function(p)
{
  p.norm <- p/sum(p)
  return (1-sum(p.norm*p.norm))
}

gini(probs)

#Just checking ...
1-(0.46*0.46+0.25*0.25+0.1*0.1+0.19*0.19)
```

Nominal Scales Association Statistics

Can we reasonably sure that modules written in COBOL are more fault-prone than modules written with C++?

	Non Faulty	Faulty
C++	231	20
COBOL	85	16

Use chi-square

- statistical tests say YES ($\alpha = 0.05$)

R: Chi-square Test

```
#Row and column names
rownames(mat) <- c("C++","COBOL")
colnames(mat) <- c("Non Faulty","Faulty")
mat


#Run chi-square statistical test
chisq.test(mat)
```

R: Complex Objects


```
#Analyze complex objects
ch <- chisq.test(mat)
ch
str(ch)
ch$p.value
ch$residuals
ch$residuals[1,2]
```

R: Chi-square test


```
#Chi-square test, but something is wrong
desharnais[,2:3]
desharnais[,c(2,3)]
desharnais[,c("V2","V3")]
teamVSleader <- desharnais[,c("V2","V3")]
tab <- table(teamVSleader)
chisq.test(tab)
tab
```

Do Clean the Data

Check if there are any

- corrupt data
- implausible values
- missing values

You may take several actions

- remove the columns with corrupt data/implausible values/missing values
- remove the rows with corrupt data/implausible values/missing values
- estimate the missing values
- ...

R: Cleaning the Dataset


```
#Cleaning the dataset
teamVSleader[teamVSleader[,1] == -1,]
teamVSleader[(teamVSleader[,1] == -1) |
    (teamVSleader[,2] == -1),]
teamVSleader[(teamVSleader[,1] != -1) &
    (teamVSleader[,2] != -1),]
good <- teamVSleader[(teamVSleader[,1] != -1) &
    (teamVSleader[,2] != -1),]
good
length(good)
length(good[,1])
#Chi-square test
goodTab <- table(good)
chisq.test(goodTab)
```

Ordinal Scales

Examples:

- (non-Computer-Science): hardness, any sort of ranking
- (Computer Science): failure criticality, subjective complexity

Values: ordered labels

Use: ordered classifications

Invariance: order among entities

- the ordering among entities is an additional piece of information over nominal scales
- we can still use whatever labels we like, provided that we know how to order them
- if we use numbers as values, the distances between two numbers have no meaning

Ordinal Scales Descriptive Statistics

All those for nominal scales, plus

- Central Tendency Indicator:
  $$median \in V \sum_{v<median} p(v) \leq 0.5 \wedge \sum_{v>median} p(v) \leq 0.5$$
- quantiles (percentiles)
- quartiles

Dispersion Indicator: interquartile range

How about the average?

Don't Use the Average with Ordinal Scales

A company needs to decide whether to use Waterfall (WF) or Agile development (AD) in the next project

Ten experts are given a questionnaire to give their advice on a 4-value scale with values "poor," "fair," "good," "excellent"

- the process with the highest average is chosen

Values	WF	AD
poor	2	3
fair	3	2
good	4	1
excellent	1	4
avg	?	?

Don't Use the Average with Ordinal Scales

The rankings are converted into "numerical" scales

Values	WF	AD
1	2	3
2	3	2
3	4	1
4	1	4
avg	2.4	2.6

Values	WF	AD
1	2	3
2	3	2
4	4	1
4.5	1	4
avg	3.15	3.1

The decision depends on the arbitrary choice of values

- the decision is arbitrary too, so why have this elaborate process to make a (deceivingly objective) arbitrary decision?

Ordinal Scales Association Statistics

We have two variables

- x: independent variable
- y: dependent variable

	Values		Ranks	
Obs.	x	y	r_x	r_y
1	43	18	3	1
2	48	27	4	3
3	12	16	1	2
4	31	29	2	4
5	80	90	6	6
6	78	40	5	5

Is there an association between them?

Ordinal Scales Association Statistics

Spearman's rho

- is Pearson's correlation coefficient, *applied to ranks*
- ranges between -1 (perfect negative association) and +1 (perfect negative association)
- $\rho = 1 - 6\frac{\sum_{i \in 1..n} d_i^2}{n^3 - n} = 1 - \frac{\sum_{i \in 1..n} d_i^2}{\frac{n^3-n}{6}}$

where

- d_i is the distance between the ranks
- n is the number of observations

Ordinal Scales Association Statistics

	Values		Ranks		Distance	
Obs.	x	y	r_x	r_y	d_i	d_i^2
1	43	18	3	1	2	4
2	48	27	4	3	1	1
3	12	16	1	2	-1	1
4	31	29	2	4	-2	4
5	80	90	6	6	0	0
6	78	40	5	5	0	0

$$\rho = 16(4+1+1+4)/(216-6) = 0.714$$

Ordinal Scales Association Statistics

Motivations
Goals
Measurement
Descriptive Analysis
Levels of Measurement
OLS
Outliers
Robust Regression
LMS
Logistic Regression
Classification
Model Validation
Final Notes

Kendall's tau: it ranges between -1 and +1

- $\tau = \frac{2(C-D)}{n^2-n} = \frac{C-D}{\frac{n(n-1)}{2}}$

where

- C is the number of concordant pairs
- D is the number of discordant pairs
- n is the number of observations

Ordinal Scales Association Statistics

	Values		Ranks		C & D	
Obs.	x	y	r_x	r_y	C	D
3	12	16	1	2	4	1
4	31	29	2	4	2	2
1	43	18	3	1	3	0
2	48	27	4	3	2	0
6	78	40	5	5	1	0
5	80	90	6	6	0	0

$\tau = 2(12 - 3)/(36 - 6) = 0.6$

R: Statistics for Ordinal Scales


```
#Compute medians
median(desharnais[,3])
median(effort)

#Compute association indicators
cor.test(good[,1], good[,2], alternative =
    "two.sided", method = "spearman")
cor.test(good[,1], good[,2], alternative =
    "two.sided", method = "spearman", exact = FALSE)
cor.test(good[,1], good[,2], alternative = "greater",
    method = "kendall")
cor.test(good[,1], good[,2], alternative = "less",
    method = "kendall")
```

Interval Scales

Examples:

- (non-Computer-Science): calendar time, centigrade temperature
- (Computer Science): milestone date

Values: numbers

Use: evaluation of distances from a "conventional" origin

Invariance: ratios of interval lengths

- $\frac{m(e_1)m(e_2)}{m(e_3)m(e_4)}$
  the distance between two entities is an additional piece of information over ordinal scales
- we can still change
  - reference origin
  - unit of measurement

Interval Scales Central Tendency Statistics

All those for ordinal scales, plus

- the average value $m = \frac{\sum_{i \in 1..n} y_i}{n}$

The average value is also the specific value of c that minimizes

- $\sum_{i \in 1..n} (y_i - c)^2$

or, equivalently, the average square residual

- $\frac{\sum_{i \in 1..n} (y_i - c)^2}{n}$

Sample variance

- $s^2 = \frac{\sum_{i \in 1..n}(y_i - m)^2}{n}$

The unbiased estimator of variance is

- $\hat{s}^2 = \frac{\sum_{i \in 1..n}(y_i - m)^2}{n-1}$

Pearson's r or Pearson product-moment correlation coefficient

- $r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$

Pearson's R^2 coefficient of determination

- $R^2 = 1 - \frac{\sum_{i \in 1..n}(y_i - \hat{y}_i)}{\sum_{i \in 1..n}(y_i - \bar{y})^2}$

R: Statistics for Interval Scales


```
#Descriptive statistics
mean(effort)
sd(effort)
```

Ratio Scales

Examples:

- (non-Computer-Science): weight measured in grams, length measured in meters
- (Computer Science): size measured by number of statements, size measured by LOC

Values: numbers

Use: evaluation of distances from a natural origin

Invariance: ratios of values

- $\frac{m(e_1)}{m(e_2)}$
- the existence of a natural origin is an additional piece of information over interval scales
- we can still change the unit of measurement as we like

Ratio Scales

Meaningful statements

- ratios of values, in particular
  - ratio of the measure of any entity to the unit of measurement

Admissible transformations:

- $m' = am$ (with $a > 0$)
  - $\frac{m'(e_1)}{m'(e_2)} = \frac{am(e_1)}{am(e_2)} = \frac{m(e_1)}{m(e_2)}$
- $m' = am + b$ (with $a > 0$)
  - $\frac{m'(e_1)}{m'(e_2)} = \frac{am(e_1)+b}{am(e_2)+b} \neq \frac{m(e_1)}{m(e_2)}$
- this is a subset of the transformations of interval scales

Statistics: all those for interval scales, plus

- (descriptive) geometric mean

Ratio Scales Dispersion Statistics

Coefficient of variation

- ratio of the standard deviation σ to the expected value μ
- $c_v = \frac{\sigma}{\mu}$

The standard deviation needs to be interpreted in the context of the expected value

Do Use Ordinal Scales Association Statistics with Ratio Measures

Are Spearman's rho and Kendall's tau useful?

- Asymptotic Relative Efficiency against Pearson parametric correlation test for bivariate normal distribution is 0.912
  - using tests on rho or tau with 1,000 observations is as efficient as using tests on Pearson's coefficient with 912

Spearman's rho is better known

Kendall's tau

- has a simpler interpretation
- approaches its asymptotic normal distribution faster than Spearman's rho

Absolute Scales

Examples:

- (non-Computer-Science): probabilities
- (Computer Science): LOC as a measure of the attribute "number of lines of code"

Values: numbers

Use: count of entities

Invariance: values

Meaningful statements: values

Admissible transformations: none!

- $m' = m$

Statistics: all those for ratio scales

Summary of Scale Types

Scale Type	Examples	Transformations
Nominal	Classifications	One-to-one
Ordinal	Preference, ranking	Monotonically increasing
Interval	Time, temperature	$m' = a\ m + b\ (a > 0)$
Ratio	Length, weight	$m' = a\ m\ (a > 0)$
Absolute	Counting	$m' = m$

Summary of Statistics

Scale Type	Central tendency	Dispersion	Dependency
Nominal	mode	H, Gini	χ^2
Ordinal	median	quantiles	ρ, τ
Interval	arith. mean	st. dev., range	Pearson's r
Ratio	geom. mean	c_v	
Absolute			

Ordinary Least Squares Regression: Assumptions

The true regression line is linear $y = \alpha x + \beta$

The values of y for any given x are

- independent
- identically normally distributed, with
  - expected value $\alpha x + \beta$
  - variance σ_e^2

Ordinary Least Squares Regression: Basic Ideas

Focus on residuals

- $res_i = y_i - \alpha \hat{y}_i = y_i - \alpha x_i - \beta$

OLS regression is based on the minimization of the average of the squared residuals

- $\underset{i \in 1..n}{avg}\left[res_i^2\right] = \frac{\sum_{i \in 1..n}(y_i - \alpha x_i - \beta)^2}{n}$

Estimation model

- $\hat{y} = ax + b$
  - α and β: true values
  - a and b: estimated values

Ordinary Least Squares Constant Regression

Special case of OLS with no variables

- $\hat{y} = m_{OLS}$

m_{OLS} is the value of parameter β that minimizes

- $\underset{i \in 1..n}{avg}\left[res_i^2\right] = \frac{\sum_{i \in 1..n}(y_i - \beta)^2}{n}$

It is well known that m_{OLS} is the arithmetic mean

- $\hat{y} = m_{OLS}$ is also used the reference case to compare OLS models with

Dispersion

Since m_{OLS} minimizes

- $\underset{i \in 1..n}{avg}[res_i^2] = \frac{\sum_{i \in 1..n}(y_i - \beta)^2}{n}$

We take this minimum value as the index of dispersion

- $s^2 = \frac{\sum_{i \in 1..n}(y_i - m_{OLS})^2}{n}$
- s^2 is the sample variance

R: Build OLS Model


```
#Linear correlation
cor.test(UFP, effort, alternative = "greater", method
    = "pearson")
effortVSUFP <- lm(effort~UFP)
str(effortVSUFP)
effortVSUFP$coefficients
summary(effortVSUFP)
coef(summary(effortVSUFP))
coef(summary(effortVSUFP))[2,4]
str(summary(effortVSUFP))
summary(effortVSUFP)$r.squared
```

Ordinary Least Squares Regression: Assumptions

We need to check if the assumptions are met

- let us check if data are distributed normally across the regression line
- we use a statistical test for normality
    - we put together all residuals
    - we use the Shapiro-Wilk test
        - H_0: the distribution of residuals is normal
        - H_1: the distribution of residuals is not normal

:

R: Statistics for Interval Scales


```
#Check the applicability of OLS
#Check the applicability of OLS
shapiro.test(effortVSUFP$residuals)

cor.test(effort, UFP, alternative = "two.sided",
    method = "spearman")
cor.test(effort, UFP, alternative = "two.sided",
    method = "spearman", exact = FALSE)
cor.test(effort, UFP, alternative = "greater", method
    = "kendall")
cor.test(effort, UFP, alternative = "less", method =
    "kendall")
```

Ordinary Least Squares Regression: Alleviating Problems

What if the assumptions are not met?

- First, we can still check if there is an association between the independent and the dependent variable

Or, we can use two typical approaches

- outlier elimination
- data transformation
  - typically a log-log transformation
    - new dependent variable is the logarithm of the old one
    - new independent variable is the logarithm of the old one

Statistical Significance OLS Degree of Correlation

OLS indicator of the degree of correlation

- $R^2_{OLS} = 1 - \frac{\frac{\sum_{i \in 1..n} (y_i - \alpha x_i - \beta)^2}{n}}{\frac{\sum_{i \in 1..n} (y_i - \mu)^2}{n}}$

Meaning

- degree of "improvement" of univariate OLS over constant OLS

True value of R in OLS is ρ_{OLS}

- $\rho_{OLS} = 0 \Leftrightarrow \alpha = 0$

Statistical Significance OLS Statistical Test

Statistic with Student's t distribution ($n-2$ degrees of freedom)

- $t = \sqrt{\frac{n-2}{1-R_{OLS}^2}} \frac{R_{OLS}}{a}(a-\alpha)$

It can be used for testing $H_0 : \alpha = 0 (\Leftrightarrow \rho_O LS = 0)$

- $t = \sqrt{\frac{n-2}{1-R_{OLS}^2}} R_{OLS}$

If $\alpha = 0$ the univariate OLS regression line coincides with the constant OLS line

- $y = \beta$

Practical Measure Validation: Do's

Building statistical model

- it shows that a measure is statistically associated with another measure
- however, this is the same as computing a multidimensional descriptive statistic

We need to use the model on a test set

- a subsequent system
- a subset of the training set
  - K-fold Cross Validation

Practical Measure Validation: Don'ts

Collecting a distribution of values for an internal measure

- characterization

Showing that a relationship exists between an internal measure X and another internal measure W

- it is useless
- it does not guarantee that X is related to a practically useful measure Y even if W is related to Y

Using the wrong kind of model

- e.g., a linear regression model to estimate fault-proneness, which is a probability, so it is between 0 and 1

Null Hypothesis Statistical Testing

Usually taken as “the null hypothesis vs. the alternative hypothesis”

- but it is not

Make sure what the null hypothesis really is

- typically the fact that a parameter is equal to 0

It is not always true that the alternative hypothesis is the one you want to prove

Why do you choose a hypothesis as the “null” hypothesis?

Statistical Significance

It is obviously very important, but it is not everything

The p-value is not the probability of obtaining a result by chance

The p-value is the probability of having obtained the result we obtained or a more extreme one, under the assumption that the null hypothesis holds

The more datapoints you have, the more likely you will find a statistically significant relationship

The statistical significance threshold is obviously arbitrary

- 0.1, 0.05, 0.001? *, **, ***?

Why not use the p-value without threshold?

Power of a Statistical Test

A chimerical concept

- hardly ever used in Empirical Software Engineering

It depends on

- statistical significance (the only thing we can always control)
- number of datapoints
- effect size (unknown)

It we knew the effect size, we probably would not even run the statistical test

Why should we aim at

- 5% statistical significance
- 80% power?

Statistical Relevance

Now, you may have found a statistically significant relationship

- it may be good for research purposes

Is it any good, practically?

- No, unless you can show a large enough effect size

Effect Size

Quantitative measure of the strength of a phenomenon

(Correlation) Effect sizes based on "variance explained"

- Pearson' correlation coefficient, eta-squared, omega-squared ...

(Difference) Effect sizes based on differences between means

- Cohen's d, Glass' Δ, Hedges' g ...

(Categorical) Effect sizes for associations among categorical variables

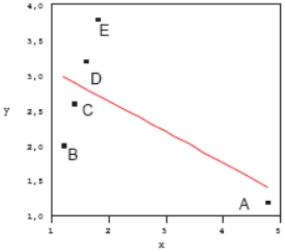
- Cohen's w, Odds ratio, Relative risk, Cohen's h, ...

They all come with (subjective) guidelines for the interpretation of values

Do Check for Outliers

An outlier

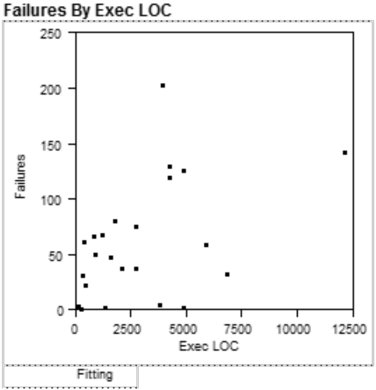
- is a data point overly influential for a regression model
- may lead an estimation model astray
- may make us believe in an incorrect model
- may make it more difficult to find a useful model



Outliers

Outliers may be due to

- rare statistical fluctuations (investigate why)
- corrupted data



Do Eliminate Outliers One-by-One Iteratively

A heuristic iterative procedure is typically used on dataset CDS

- by removing outliers one—the farthest one—at a time
- based on some
  - outlier criterion
  - distance function
  - distance threshold

1: COS := out(CDS, distanceFunction, distanceThreshold)
   //COS ⊆ CDS is the set of outliers
2: **while** COS ≠ ∅ **do**
3:   odp := farthest(COS, CDP, distanceFunction)
     //odp ∈ COS is the farthest outlier for CDS
     CDS := CDS - {odp}
     //Remove odp from CDS
4: **end while**

Don't Eliminate Sets of Outliers Wholesale

Ineffective way of eliminating outliers

```
COS := out(CDS, distanceFunction, distanceThreshold)
while COS ≠ ∅ do
    CDS := CDS - COS
end while
```

Cook's Distance

Cook's distance evaluates the influence on predictions due to a single data point z in an $(v+1)$-dimensional space

- v independent variables
- 1 dependent variables

Definition of Cook's distance for datapoint z

- $Cook(z) = \frac{\sum_{i \in 1..n}(\hat{y}_{i,CDS} - \hat{y}_{i,CDS-\{z\}})^2}{par \cdot MSE}$
- $\hat{y}_{i,CDS}$ is the i-th predicted value with the entire dataset
- $\hat{y}_{i,CDS-\{z\}}$ is the i-th predicted value when point z is removed
- par is the number of parameters in the model
- MSE is the Mean Square Error of the model

Thresholds

- 1, 4/v, 4/(n - v - 1), ...

Mahalanobis Distance

The Mahalanobis distance is

- the distance of z from the $(v+1)$-dimensional center of mass m
- divided by the width in the direction of z of the ellipsoid that best represents the data set's probability distribution

The idea is that sheer distance of z from m is not good enough

- just because z is far from m does not mean that z is an outlier
  - if z is far from m, but "close enough" to the regression variety (i.e., in the ellipsoid), then it is not an outlier
  - if z is not far from m, but "far enough" from the regression variety, then it is an outlier

- $Mahalanobis(z) = \sqrt{(z-m)S^{-1}(z-m)}$

where S is the covariance matrix of the dataset

Thresholds are given in terms of the F-distribution

To Jackknife or not to Jackknife?

The distance of z can be computed in two ways

- from the centroid of the entire CDS, or
- from the centroid of CDS - $\{z\}$

The second distance uses a jackknife procedure

- it further removes the influence of z in biasing the position of the centroid
- it is more complex to implement and compute
- it is better for eliminating outliers

Practical Advice

Set a maximum percentage of outliers

- you may end up with most of the data points that are classified as outliers

Accept the result you obtain

- removing outliers may very well play "against you"
  - you may have a great correlation on the entire data set
  - you may obtain a much worse correlation after you have removed one or more outliers
- the aim of the game is not to obtain a great correlation, but to obtain a valid, useful model

R: Cook's Distance

```
#Cook's distance
effortVSUFP = lm(effort~UFP)
cooksDistances <- cooks.distance(effortVSUFP)
cooksDistances
cooksDistances > 4/length(cooksDistances)
cooksDistances > 1
sum(cooksDistances > 4/length(cooksDistances))
sum(cooksDistances > 1)
```

R: Cook's Distance


```
removeOutliersCook4l <- function(x,y) {
  done = FALSE
  while ( !done ) {
    model = lm(y ~ x)
    cooksDistances <- cooks.distance(model)
    maxCooksDistance = max(cooksDistances)
    l = length(y)
    if(maxCooksDistance < 4/l) { done = TRUE; break; }
    removeNext = which.max(cooksDistances)
    y = y[-c(removeNext)]
    x = x[-c(removeNext)]
  }
  return (list(x,y))
}

nonOutliersCook4l <- removeOutliersCook4l(loc, effort)
```

R: Cook's Distance


```
removeOutliersCook1 <- function(x,y) {
  done = FALSE
  while ( !done ) {
    model = lm(y ~ x)
    cooksDistances <- cooks.distance(model)
    maxCooksDistance = max(cooksDistances)
    l = length(y)
    if(maxCooksDistance < 1) { done = TRUE; break; }
    removeNext = which.max(cooksDistances)
    y = y[-c(removeNext)]
    x = x[-c(removeNext)]
  }
  return (list(x,y))
}
nonOutliersCook1 <- removeOutliersCook1(loc, effort)
nonOutliersCook1
```

Do Use Robust Statistics

Take a data set with n points

Take an estimator

How many of those data points need to be corrupted to lead the estimator astray?

To find out

- move k of them towards infinity
- if the estimator does not move towards infinity, the max value of k/n is an indication of the robustness of the estimator

Robustness of the average and of the estimators of the parameters of Ordinary Least Square (OLS) regression: 1/n

- one corrupted data point may be enough

Robust Statistics

Robustness of the median: 50

- up to 50% of the data points may be corrupted, still the median does not go to infinity

Robust statistics are like the median

- up to 50% robustness
- not more than 50%
  - otherwise we cannot tell the "good" data points from the "bad" data points

Least Median of Squares: Assumptions

The true regression line is linear $y = \alpha x + \beta$

The values of y for any given x are

- independent
- identically distributed

No assumptions about expected value and variance

Least Median of Squares Regression: Basic Ideas

LMS regression is based on the minimization of the median of the squared residuals

- $\underset{i \in 1..n}{med}[res_i^2] = \underset{i \in 1..n}{med}[(y_i - \alpha x_i - \beta)^2]$

Since the squared residuals are ordered like the absolute values, we can equivalently minimize

- $\underset{i \in 1..n}{med}[|res_i|] = \underset{i \in 1..n}{med}[|y_i - \alpha x_i - \beta|]$

Least Median of Squares (LMS) regression is a robust regression technique, originally introduced by Rousseeuw and Leroy from the University of Antwerpen

LMS: Basic Ideas

We use the low median

Let us order absolute residuals in ascending order

- we want to minimize the value of $|res_{\lceil \frac{n}{2} \rceil}|$, the residual in the middle

We do not care about how big the residuals with $i > \lceil \frac{n}{2} \rceil$ may be, so

- approximately half of the residuals may be as big as possible, still the value of $|res_{\lceil \frac{n}{2} \rceil}|$ will not change, so
  - LMS is as robust as possible

Constant LMS Regression Properties

Special case of LMS with no variables

- $\hat{y} = m_{LMS}$

m_{LMS} is the value of parameter β that minimizes

- $\underset{i \in 1..n}{med}[|res_i|] = \underset{i \in 1..n}{med}[|y_i - \beta|]$

m_{LMS} is a new, robust indicator of interval central tendency in its own right

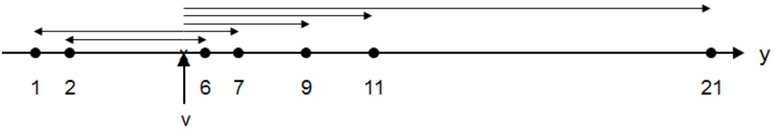
- $\min\{V_Y\} \leq m_{LMS} \leq \max\{V_Y\}$ (Cauchy's property)
- $m_{LMS}\{ay_i + b\} = a \cdot m_{LMS}\{y_i\} + b$

We also use $\hat{y} = m_{LMS}$ as the reference case to compare LMS models

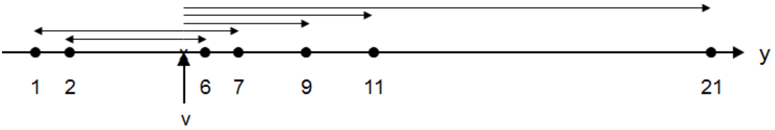
Constant LMS Regression

m_{LMS} is the midpoint of the narrowest interval that contains at least $\lceil \frac{n}{2} \rceil$ data points

- median distance of data points from a point v is computed as maximum distance of closest $\lceil \frac{n}{2} \rceil$ data points to the point



Constant LMS Regression



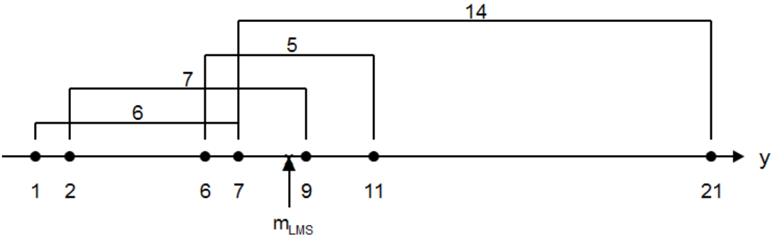
Take $X = 5.5$

Point	Distance
1	4.5
2	3.5
6	0.5
7	1.5
9	3.5
11	5.5
21	15.5

Point	Distance
6	0.5
7	1.5
2	3.5
9	3.5
1	4.5
11	5.5
21	15.5

Given any interval, the point that minimizes the maximum distance to the points in the interval is the midpoint

- m_{LMS} is the midpoint of the narrowest interval that contains at least $\lceil \frac{n}{2} \rceil$ data points



Dispersion

Since m_{LMS} minimizes

- $\underset{i \in 1..n}{med}[|res_i|] = \underset{i \in 1..n}{med}[|y_i - \beta|]$

We take this minimum value as the index of dispersion

- $mar = \underset{i \in 1..n}{med}[|y_i - m_{LMS}|]$

Univariate LMS Regression

The LMS regression line lies halfway in the narrowest strip (distance measured along the y-axis) that encloses at least $\lceil \frac{n}{2} \rceil$ data points

Statistical Significance LMS Degree of Correlation

Degree of "improvement" of univariate LMS over constant LMS

$$R^2_{LMS} = 1 - \frac{med_{i \in 1..n}\{|y_i - ax_i - b|\}}{med_{i \in 1..n}\{|y_i - \bar{y}_{LMS}|\}}$$

where $\bar{y}_{LMS}$ is m_{LMS} for the $\{y_i\}$ data set

$$S^2_{LMS} = 1 - \frac{med_{i \in 1..n}\{(y_i - ax_i - b)^2\}}{med_{i \in 1..n}\{(y_i - \bar{y}_{LMS})^2\}} \quad (1)$$

It can be shown that it is always $R^2_{LMS} \leq S^2_{LMS}$

- same statistical inference method for R^2_{LMS} and S^2_{LMS}
- we use R^2_{LMS}

True value of R_{LMS} is ρ_{LMS}

- $\rho_{LMS} = 0 \Leftrightarrow \alpha = 0$

Statistical Significance LMS Statistical Test

If $\alpha = 0$ the univariate LMS regression line coincides with the constant LMS line

- the distributions of absolute residuals from univariate LMS regression line and constant LMS line coincide
- the median of their difference is null

We test if the median of their difference is null

- nonparametric test
- Fisher's sign test's statistic: we can use
  - exact test, or
  - asymptotic approximation

Data Set desharnais1_1_1 (PROMISE Data Set)

Descriptive statistics

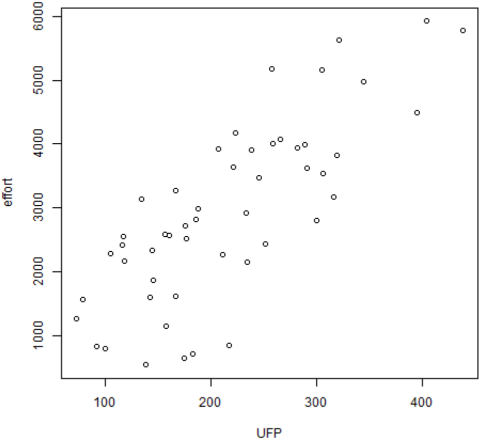
	n	min	max	med	m_{OLS}	σ_{OLS}	m_{LMS}	mar_C
Effort	81	546	23940	3647	5046.31	4419.77	2786	1386
Trans.	81	9	886	140	182.12	144.04	93	53
Entities	81	7	387	99	122.33	84.88	65	34
UFP	81	73	1127	266	304.46	180.21	239	82

Data Set desharnais1_1_1 OLS Results

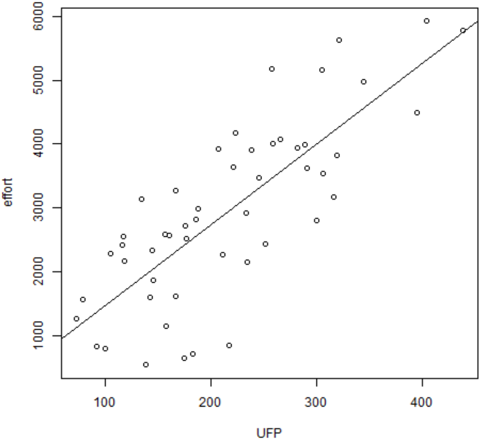
Effort is the dependent variable

Variable	n	a	b	p	R^2_{OLS}	w
UFP	81	17.30	-220.08	< 0.0001	0.50	< 0.0001
UFP	50	12.64	211.98	< 0.0001	0.50	0.3276
Trans.	81	17.85	1795.19	< 0.0001	0.34	< 0.0001
Trans.	58	15.26	1251.84	< 0.0001	0.20	0.2920
Entities	81	26.57	1796.33	< 0.0001	0.26	< 0.0001
Entities	53	34.88	217.46	< 0.0001	0.43	0.0130

Effort vs. UFP after Removal of Outliers

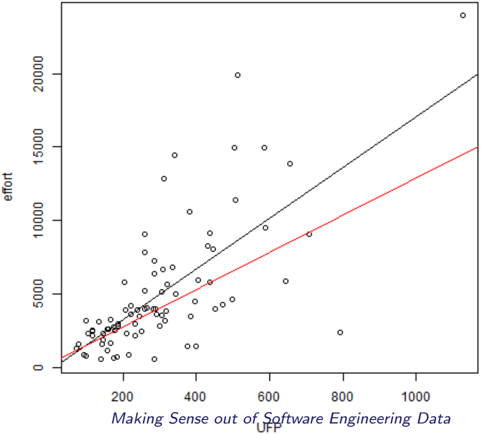


Regression Line after Removal of Outliers



OLS and LMS Regression Lines

- Before removal: black line
- After removal: red line



Data Set desharnais1_1_1 LMS Results

Effort is the dependent variable

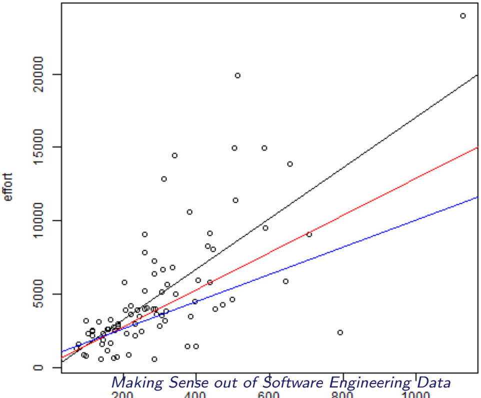
Variable	a	b	p	mar_U	R^2_{LMS}	S^2_{LMS}
UFP	9.27	803.27	¡0.0001	925.04	0.33	0.556
Trans.	7.29	2239.56	0.0027	1111.40	0.20	0.36
Entities	9.21	1677.13	0.0174	1150.24	0.17	0.31

R: LMS


```
#LMS regression
#load package MASS
lms.effortVSUFP <- lmsreg(UFP, effort)
lms.effortVSUFP
summary(lms.effortVSUFP)
str(lms.effortVSUFP)

#Compute LMS unidimensional central tendency indicator
lqs(effort~effort, method = "lms")
```

OLS Regression Lines

- Before removal: black line
- After removal: red line
- LMS: blue line



R: Statistical Significance of LMS Models


```
#Step 1: compute m_LMS, the LMS unidimensional central
    tendency indicator
m_LMS <- lqs(effort~effort, method = "lms")
m_LMS
str(m_LMS)
m_LMS$coefficients

#Step 2: compute reg_LMS, the LMS regression line
reg_LMS <- lqs(effort~UFP, method = "lms")
reg_LMS
str(reg_LMS)
reg_LMS$coefficients
reg_LMS$fitted.values
```

R: Statistical Significance of LMS Models


```
#Step 3: compute absRes0, the vector of the absolute
    residuals of the data from m_LMS: use function
    abs(x)
absRes0 <- abs(effort-m_LMS$fitted.values)

#Step 4: compute absRes1, the vector of the absolute
    residuals of the data from reg_LMS
absRes1 <- abs(effort-reg_LMS$fitted.values)

#Step 5: compute g, the number of times a residual in
    absRes0 is greater than the corresponding residual
    in absRes1
diff <- absRes0 - absRes1
length(diff[diff>0])

#Step 6: compute binom.test(g, length(effort))
binom.test(length(diff[diff>0]),length(effort))
```

Binary Logistic Regression

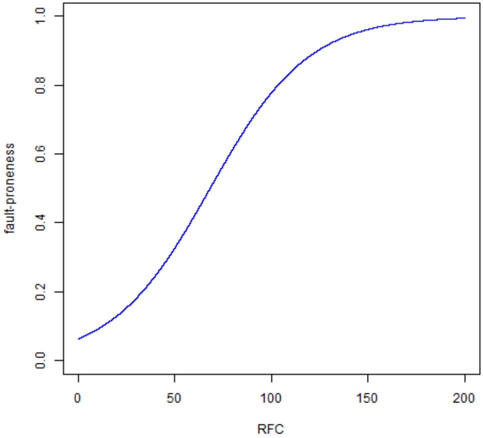
Logistic Regression is a probability estimation technique

$$P(Y = Positive|\underline{X} = \underline{x}) = \frac{e^{c_0 + c_1 x_1 + \ldots + c_v x_v}}{1 + e^{c_0 + c_1 x_1 + \ldots + c_v x_v}}$$

Response variable

- in Binary Logistic Regression, the response variable is a *nominal* measure Y can only take two values, *Negative* and *Positive*
  - Logistic Regression provides the probability that $Y = Positive$, for given values of the independent variables

Model



Logistic Regression

S-shaped curve between a given X_i and P (all other X_j's are constant) is

- very steep: X_i has a very large impact on Y
  - when the curve is step-like, there is perfect separation between negative and positive estimates
- quite flat: X_i has a small impact on Y, and is not useful for classification
  - when the curve is totally flat, there is no impact of X_i on Y

Special case of Binary Logistic Regression

- constant model, i.e., without variables
- $P(Y = Positive|underlineX = \underline{x}) = \frac{AP_{tr}}{n_{tr}}$

Logistic Regression

Coefficients are estimated via Maximum Likelihood Estimation

- with usual assumption of independence of observations

Existence of impact of X_i is assessed via the p-value of c_i

Goodness-of-fit: proportion of log-likelihood explained by the BLR model (ranges between 0 and 1): high values are rare

- $R^2_{BLR} = \frac{LL_0 - LL}{LL_0}$
- for technical reasons, high values are rare
- LL: log-likelihood of BLR model
- LL_0: log-likelihood of BLR model without variables
  - $LL_0 = AN_{tr} \log \frac{AN_{tr}}{n_{tr}} + AP_{tr} \log \frac{AP_{tr}}{n_{tr}}$

R: Logistic Regression


```
#Upload a dataset
mc2 <-
    read.csv(file="j:\\RData\\mc2.txt",head=TRUE,sep=",")
mc2

#Extract response variable
faulty <- mc2[40][[1]]
faulty

#Plot the response variable: not very good
hist(faulty)

#Plot the response variable: this is better
hist(faulty, breaks = 2)
```

R: Logistic Regression

```
#Plot the response variable: not useful
boxplot(faulty)

#Extract independent variable loc
loc <- mc2[1][[1]]

#Plot the response variable: this is better
boxplot(LOC_BLANK~Defective, mc2)

plot(loc, faulty)

#Show histograms
locNonFaulty <- loc[faulty %in% 0]
hist(locNonFaulty, breaks = 20)
locFaulty <- loc[faulty %in% 1]
hist(locFaulty, breaks = 20)
```

R: Logistic Regression


```
#Build univariate Binary Logistic Regression model
faultyVSloc <- glm( faulty~loc,
    family=binomial(link="logit"))
faultyVSloc

summary(faultyVSloc)

str(summary(faultyVSloc))

coef(summary(faultyVSloc))
```

R: Logistic Regression


```
#Find the range of the independent variable
range(loc)

#Plot the Binary Logistic Regression model
xweight <- seq(min(loc), max(loc)*1.1, 0.1)
yweight <- predict(faultyVSloc, list(loc =
    xweight),type="response")

plot(loc, faulty, pch = 16, xlab = "loc", ylab =
    "faulty")
lines(xweight, yweight)
```

R: Logistic Regression


```
#Plot thresholds on the model
faultyProportion <- sum(faulty)/length(faulty)
faultyProportion

abline(h=faultyProportion)
coef(faultyVSloc)
x <- (log(faultyProportion/(1-faultyProportion))-
         coef(faultyVSloc)[1])/coef(faultyVSloc)[2]
abline(v = x)
```

R: Logistic Regression


```
#Build univariate Binary Logistic Regression model
    with bc as independent variable
bc <- mc2[2][[1]]
boxplot(BRANCH_COUNT~Defective, mc2)

faultyVSbc <- glm(faulty~bc,
    family=binomial(link="logit"))
faultyVSbc
summary(faultyVSbc)

#Build multivariate Binary Logistic Regression model
    with loc and bc as independent variables
faultyVSlocbc <- glm(faulty~loc+bc,
    family=binomial(link="logit"))
summary(faultyVSlocbc)
```

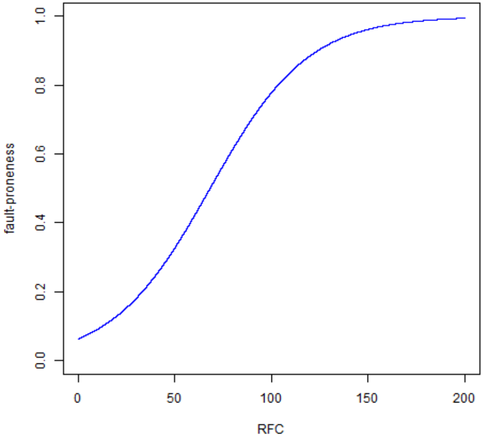
Classification

A fault-proneness model estimates the *probability* that a software module is faulty
That is not good enough in practice, because practitioners

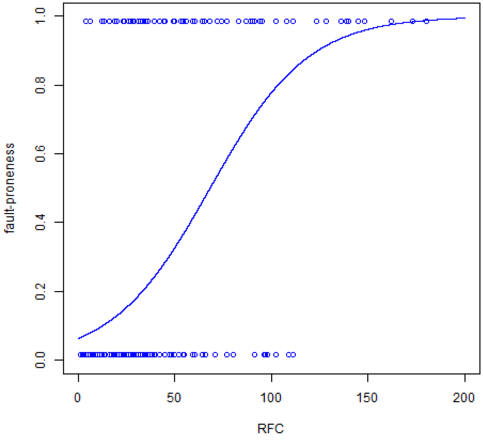
- need to classify modules as fault-prone and not-fault-prone
- need to know the safe ranges for measures
- may need to find safe, acceptable, and unsafe ranges

We need thresholds in addition to fault-proneness models

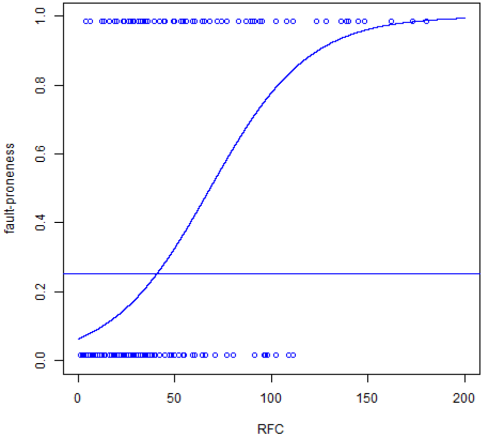
Model



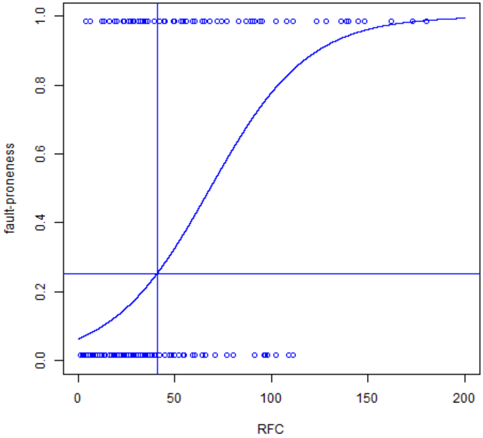
Test Set



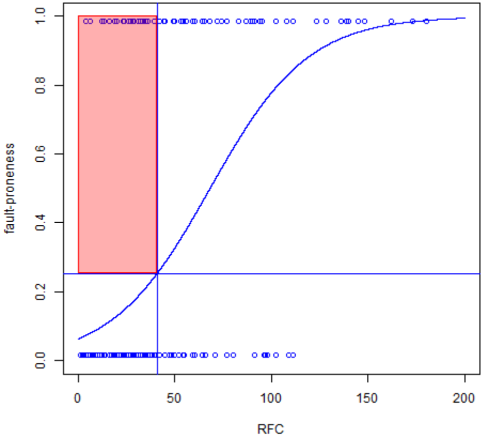
Threshold on Fault-proneness



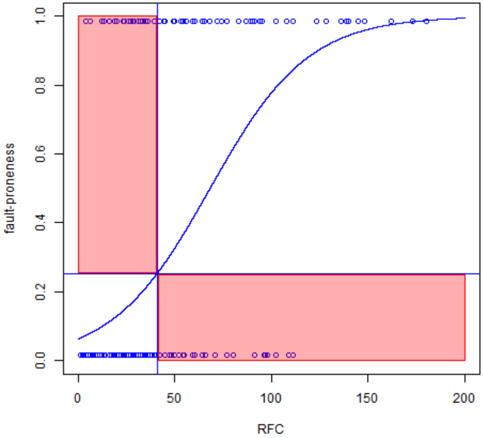
Threshold on Independent Variable



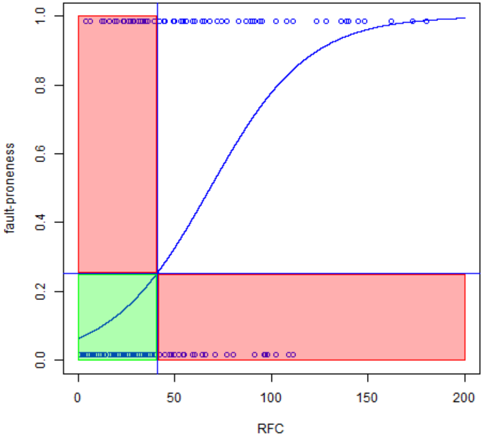
False Negatives



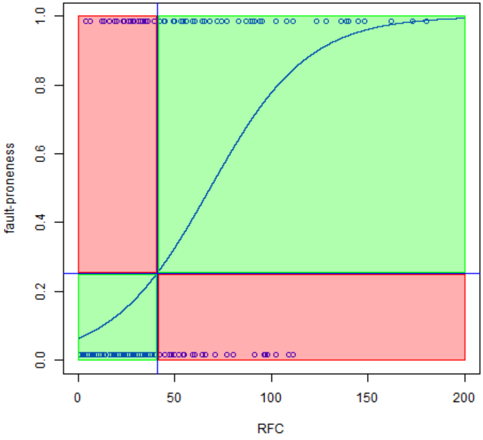
False Positives



True Negatives



True Positives



Contingency Table

		Actual		
		Non-faulty	Faulty	
Estimated	Non-faulty	TN	FN	EN
	Faulty	FP	TP	EP
		AN	AP	n

Classification Thresholds: Do's

Choose meaningful thresholds

- typically data-dependent

Examples

- $t_{tr} = AP_{tr}/n_{tr}$: *known* proportion of faulty modules in the training set
  - probability of picking a faulty module in the training set at random, i.e., without any further information about the module
- $t_{ts} = AP_{ts}/n_{ts}$: *unknown* proportion of faulty modules in the test set
- $t_{all} = AP_{all}/n_{all}$: *unknown* proportion of faulty modules in the entire data set

Classification Thresholds: Don'ts

Data-independent thresholds

Example

- $t = 0.5$: a theoretical threshold, used for no prior knowledge

Researcher-dependent thresholds

- any threshold that will get me good results
- e.g., an unreasonably low fault-proneness threshold, so all modules are classified fault-prone
  - all actually faulty modules are included and Recall $= 1$!
    - but it's not even necessary to make all of this effort to get this result . . .

Accuracy Indicators

Precision: proportion of estimated positives that are actually positive

$$Precision = \frac{TP}{EP}$$

Recall: proportion of actual positives that are also estimated as positives

$$Recall = \frac{TP}{AP}$$

Accuracy: proportion of correct classifications

$$Recall = \frac{TP + TN}{n}$$

Precision vs. Recall

Ideally, both Precision and Recall should be maximized

- contrasting goals

Trivial decision criterion: estimate all datapoints positive

- $t = 0$
- $Recall = 1$, $Precision = ?$

Other decision criterion

- we may miss some true positives but
- detect some true negatives
- have some false negatives
  - Precision may go up (fewer false positive detected), but recall may go down

We need to make a decision based on the risk of errors

Accuracy Indicators

F-measure: harmonic mean of *Precision* and *Recall*

$$FM = \frac{2}{\frac{1}{Precision} + \frac{1}{Recall}}$$

Weighted F-measure: weighted harmonic mean of *Precision* and *Recall* ($w \in [0, 1]$)

$$FM(w) = \frac{1}{\frac{w}{Precision} + \frac{1-w}{Recall}}$$

Accuracy Indicators

ϕ: quantifies the degree of association in 2 $\times$ 2 tables

$$\phi = \frac{TP \cdot TN - FP \cdot FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} = \sqrt{\frac{\chi^2}{n}}$$

also known as Matthews' Correlation Coefficient

For r $\times$ c tables, use Cramer's V (where $k = \min\{r, c\}$)

$$V = \sqrt{\frac{\chi^2}{n(k-1)}}$$

- $\phi = V$ for 2 $\times$ 2 tables
- ϕ and V are statistically well-founded
- ϕ and V come with statistical tests, based on χ^2

Tree-building Algorithms

A classification tree allows the prediction of the value

- of one variable (the dependent variable) of an instance, based on
- the values of other variables of the instance (the independent variables)

We consider a binary dependent variable Y (e.g., the presence of faults in a software module) with values

- $Y = 0$: e.g, no faults in a module, and
- $Y = 1$: e.g., at least one fault in the module

ID3

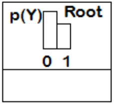
ID3 can be used only with discrete independent variables:

- nominal variables, e.g., programming language
- ordinal variables, e.g., failure severity

ID3 uses the independent variables to recursively build a classification tree that can be used to classify new instances

At the beginning of the process, we use the entire set of instances to classify a new instance, based on the probability distribution of the independent variable Y

Building a Classification Tree with ID3



ID3

The independent variables may help classify the instances

- for example, modules written in one programming language may be more error-prone than the others

Given a set of independent variables $(X, W, Z, \ldots)$, we choose first the one that provides the largest reduction in some dispersion/uncertainty figure of merit

Figures of Merit

Information gain $H(Y) - H(Y|X)$, with

- $H(Y) = -\sum_{y \in V_Y} p(y)\ log_2 p(y)$
- $H(Y|x) = -\sum_{y \in V_Y} p(y|x)\ log_2 p(y|x)$
- $H(Y|X) = \sum_{y \in V_Y} p(x) H(Y|x)$

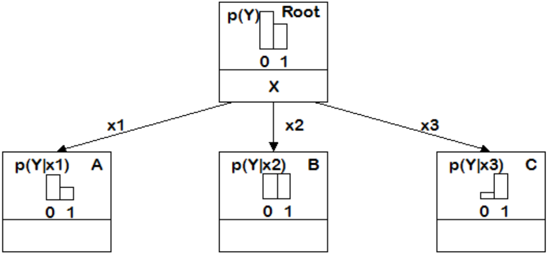
Information gain ratio $\frac{H(Y) - H(Y|X)}{H(X)}$

- to account for the fact that independent variables with more values tend to provide larger information gains

Average Gini

- $\underset{x \in V_X}{avg}\,[Gini_x] = 1 - \sum_{y \in V_Y} p^2(y|x)$

Building a Classification Tree with ID3



ID3

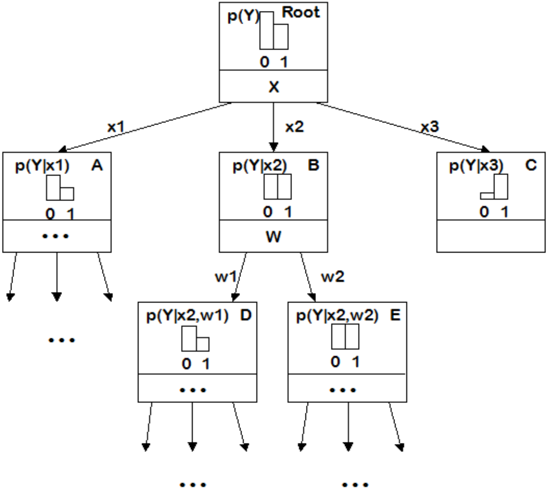
ID3 recursively builds a subtree from each node, until

- the uncertainty reduction obtained with any independent variable is below a specified threshold, or
- the number of instances associated with the node is below a specified threshold

to reduce the risk of overfitting

Each node is associated with a conditional probability distribution: it is conditional on the values of the attributes on the path from the root down to the node

Building a Classification Tree with ID3



ID3: Stopping Criteria

ID3 recursively builds a subtree from each node, until

- the uncertainty reduction obtained with any independent variable is below a specified threshold, or
- the number of instances associated with the node is below a specified threshold

to reduce the risk of overfitting

Each node is associated with a conditional probability distribution: it is conditional on the values of the attributes on the path from the root down to the node

Bayes Classifiers

Bayes Classifiers are based on a property of probabilities of events

- $p(A, B) = p(A|B)p(B) = p(B|A)p(A)$

We use

- event A: $Y = y$
- event B: $X_1 = x_1, \ldots, X_v = x_v$

Here's Bayes' Theorem

$$p(Y = y|X_1 = x_1, \ldots, X_v = x_v) = \frac{p(X_1 = x_1, \ldots, X_v = x_v|Y = y)}{p(X_1 = x_1, \ldots, X_v = x_v)} p(Y = y)$$

Prior Distribution

$p(Y = y)$ is the probabilistic knowledge that $Y = y$ based before getting more information, that is, on the entire population

- probability that a software component is $Y = \mathit{Faulty}$, without any further information on the component

Prior Distribution

The prior distribution is the crux of all Bayesian approaches

- how do we know it?

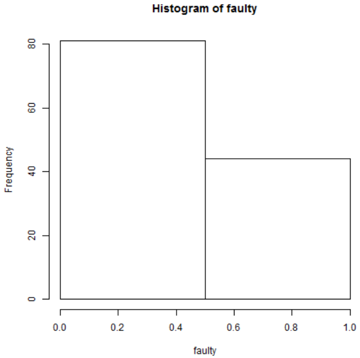
Different strategies

- equiprobable distribution
  - it reflects no previous knowledge
  - $p(Y = Faulty) = 0.5 = p(Y = Nonfaulty)$
- Maximum Likelihood Estimation for the probability of each event based on the training set
  - $p(Y = y) = \frac{observationsforwhichY = y}{n}$
  - $p(Y = Faulty) = 0.2$ and $p(Y = Nonfaulty) = 0.8$

Estimate Prior Distribution

Use Y data, so

- $p(Faulty) = \frac{AP}{n}$
- $p(NonFaulty) = \frac{AN}{n}$



Posterior Distribution

$p(Y = y | X_1 = x_1, \ldots, X_v = x_v)$ is the probabilistic knowledge that $Y = y$ after getting more information on the component, that is, on a specific subpopulation

- probability that a software component is $Y = \mathit{Faulty}$, when it is known that it is $X_1 = \mathit{interface\_component}$, written in $X_2 = \mathit{Java}$, by $X_3 = \mathit{inexperienced\_programmer}$
- this is the value that we want to compute

Likelihood

$p(X_1 = x_1, \ldots, X_v = x_v | Y = y)$ is the probability of obtaining $X_1 = x_1, \ldots, X_v = x_v$ when $Y = y$, that is, by restricting the sample

- probability that a software component is $X_1 = \mathit{interface\_component}$, written in $X_2 = \mathit{Java}$, by $X_3 = \mathit{inexperienced\_programmer}$, given that is $Y = \mathit{Faulty}$
- this is obtained via a model

Evidence

$p(X_1 = x_1, \ldots, X_v = x_v | Y = y)$ is the probabilistic knowledge that $X_1 = x_1, \ldots, X_v = x_v$ on the entire population

- probability that a software component is $X_1 = \textit{interface\_component}$, written in $X_2 = \textit{Java}$, by $X_3 = \textit{inexperienced\_programmer}$, regardless of the fact that it is or it is not faulty
- how can we know it?
  - luckily enough, it does not matter

Decision Procedure: Prior

If I have to bet on the outcome of an experiment before getting any additional information, it would be rational to bet on the outcome with the highest probability according to the prior distribution

- value $Y = y$ that maximizes $p(Y = y)$
- if $p(Y = Faulty) = 0.2$ and $p(Y = NonFaulty) = 0.8$, it is rational to say that
  - any component taken at random is nonfaulty
  - any new component (assuming it is from the same population) is non-faulty

Decision Procedure: Posterior

If I have to bet on the outcome of an experiment after getting more information, it would be rational to bet on the outcome with the highest probability according to the posterior distribution

- value $Y = y$ that maximizes $p(Y = y | X_1 = x_1, \ldots, X_v = x_v)$
- if $p(Y = Faulty | X_1 = interface\_component, X_2 = Java, X_3 = inexperienced\_programmer) = 0.6$ and $p(Y = Nonfaulty | X_1 = interface\_component, X_2 = Java, X_3 = inexperienced\_programmer) = 0.4$, it is rational to say that
  - any component taken at random is faulty
  - any new component (assuming it is from the same population) is faulty

Decision Procedure: Evidence

It is not necessary to know the evidence $p(X_1 = x_1, \ldots, X_v = x_v | Y = y)$, because

- it does not depend on the specific outcome
- it is a proportionality factor

We take the value Y = y that maximizes $p(Y = y | X_1 = x_1, \ldots, X_v = x_v | Y = y) \propto p(Y = y | X_1 = x_1, \ldots, X_v = x_v | Y = y) p(Y = y)$

Naïve Bayes Classifiers

We need to know the joint distribution of $X_1 = x_1, \ldots, X_v = x_v$ when $Y = y$ In general (Bayes Theorem again!)

$$p($$

$$p(X_1 = x_1|Y = y)p(X_2 = x_2,$$

$$p(X_1 = x_1|Y = y)p(X_2 = x_2|Y = y, X_1 = x_1)p(X_3 = x_3, \ldots, X_v =$$

$$p(X_1 = x_1|Y = y)p(X_2 = x_2|Y = y, X_1 = x_1) \ldots p(X_v = x_v|Y = y$$

How can we know all of these conditional probability distributions?

Approximation: conditional independence among the X's

$$p(X_1 = x_1, \ldots, X_v = x_v | Y = y) =$$
$$p(X_1 = x_1 | Y = y) p(X_2 = x_2 | Y = y) \ldots p(X_v = x_v | Y = y)$$

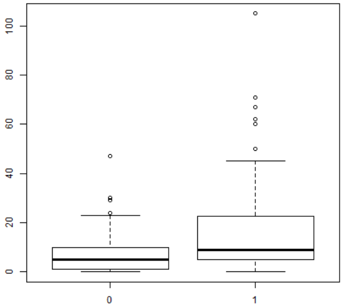
For each distribution, we build a model

- binomial, Poisson, normal, Gamma
  - $p(X_1 = interface\_component | Y = Faulty)$
  - $p(X_2 = Java | Y = Faulty)$
  - $p(X_3 = inexperienced\_programmer | Y = Faulty)$

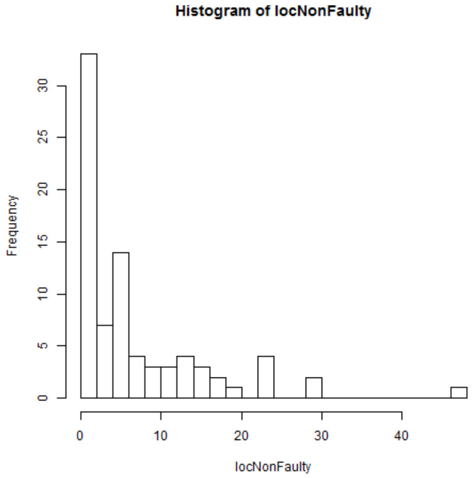
Estimate Posterior Distribution

Model $\underline{x}$ when y is known

- binomial, Poisson, normal, Gamma



When $Y = NonFaulty$



Validation

Building good models requires

- model selection
- performance estimation

In model selection, we need to identify

- the functional form
- the "optimal" parameter(s)

In performance estimation, we need to estimate how well it will perform

- usually, the true error rate, i.e., the classifier's error rate on the entire population

Validation

Validation would be perfect if we knew the entire population

- but why estimate anything, then?

In real life, we have a sample

How do we use it best?

Validation

First approach

- use the entire sample to train the classifier (= build the model)
- estimate the error rate

Two fundamental problems

- the final classifier is tailored on the sample and will typically overfit it
  - especially classifiers with lots of parameters
- the error rate estimate will be optimistic (lower than the true error rate)
  - one may very well have 100% correct classification on training data

Validation

Ideally, we should use the model obtained on a data set on a set of data points from subsequent projects

- this is not always possible

A practically useful approach split the available data into disjoint subsets

- the holdout method

Split dataset into two groups

- Training set: used to train the classifier
- Test set: used to estimate the error rate of the trained classifier

Validation

Two basic problems

- in small datasets, we cannot afford leaving out some of the dataset for testing the classifier later on
- there is randomness in selecting the training and the testing set
  - what if we are not lucky in splitting the dataset?

Techniques exist to deal with these problems

- Cross Validation
- Random Subsampling
- K-Fold Cross-Validation
- Leave-one-out Cross-Validation

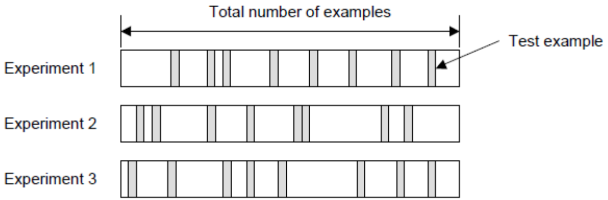
They require more computations than a one-shot holdout

Random Subsampling

Perform k data splits of the dataset

- at each data split i
- randomly select a (fixed) number of observations
- retrain the classifier from scratch with the training observations
- estimate the error E_i with the test observations

The true error estimate is obtained as the average of the separate estimates E_i



K-fold Partitioning

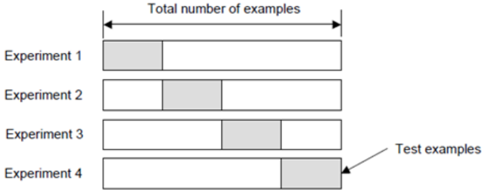
Create a K-fold partition of the dataset

- for each of the K experiments, use K-1 folds for training and the remaining one for testing

K-fold Cross validation is similar to Random Subsampling

- advantage: all the observations in the dataset are eventually used for both training and testing

The true error is estimated as the average error rate

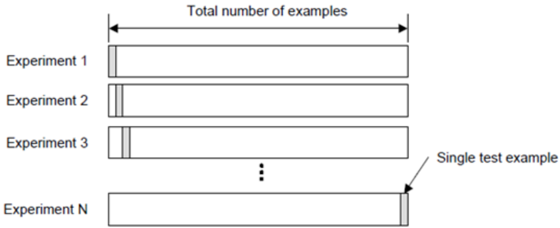


Leave-one-out Cross Validation

It is a special case of K-fold Cross Validation

- with K = N, equal to the total number of examples
- for a dataset with N examples, perform N experiments
- for each experiment, use N-1 examples for training and the remaining example for testing

The true error is estimated as the average error rate



Choosing the Number of Folds

By increasing the number of folds, also increase

- the accuracy of the true error rate estimator
- the variance of the true error rate estimator
- the computation time

The choice of the number of folds depends on the size of the dataset

- for large datasets, even 3-fold Cross Validation will be quite accurate
- for small datasets, we may have to use leave-one-out in order to train on as many examples as possible

Common choice for K-fold Cross Validation: K=10

Do Examine Threats to Validity

Dealing with the threats to validity is what we have been trying to do all along

- Internal Validity
- External Validity
- Construct Validity

Do Show Negative Results

The set of published studies is clearly biased

- only the studies with a happy ending are usually published

This does not make much sense

- knowing that something does not work may be even more important that knowing that something else works

Do not hide your negative results