Statistics with R

Mining Software Repositories 2015
University of Koblenz

Presenter: Hakan Aksu

Content

1. Introduction to R
2. Basic Statistics
3. Advanced Statistics
4. Example (MSR-Papers)

INTRODUCTION TO R

What is R?

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. [1]

- R has
  - an effective data handling and storage facility,
  - a suite of operators for calculations on arrays, in particular matrices,
  - a large, coherent, integrated collection of intermediate tools for data analysis,
  - graphical facilities for data analysis and display either directly at the computer, and
  - a well developed, simple and effective programming language (called 'S') which includes conditionals, loops, user defined recursive functions and input and output facilities. [1]

How to get R?

- Download and install the free statistic-software-package R from

http://www.r-project.org/

- R IDE: RSTUDIO

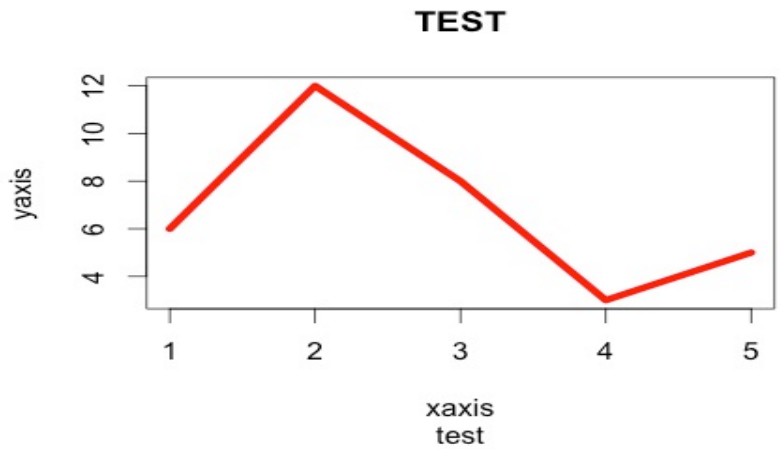
http://www.rstudio.com/

Simple Manipulations in R

```
a <- “Hello World!“                # Assign String to variable
b <- 3                             # Assign Integer to variable
c <- paste(“s1“,...,“sn“,sep=“-“) # Concatination of Strings with Seperator

d <- c(3,5,7,9,11)                 # Define vector and assign to variable
d[2]                               # 2nd value of the vector d
d[-2]                              # Vector d without 2nd value
e <- c(1:5)                        # e <- c(1,2,3,4,5)

f <- matrix(c(1:12),ncol=3)        # Define Matrix with 3 columns
g <- cbind(e,d)                    # Define Matrix with two columns e and d
```

Graphics in R
plot

```
plot(g)                          # draw graphic – e as x-axis and d as y-axis
vx <- c(1:5)                     # Define vector vx
vy <- c(6,12,8,3,5)              # Define vector vy

plot(vx,vy,
type="l“, col="red", lwd=6,
sub="test", main="TEST",
xlab="xaxis",ylab="yaxis“)
```

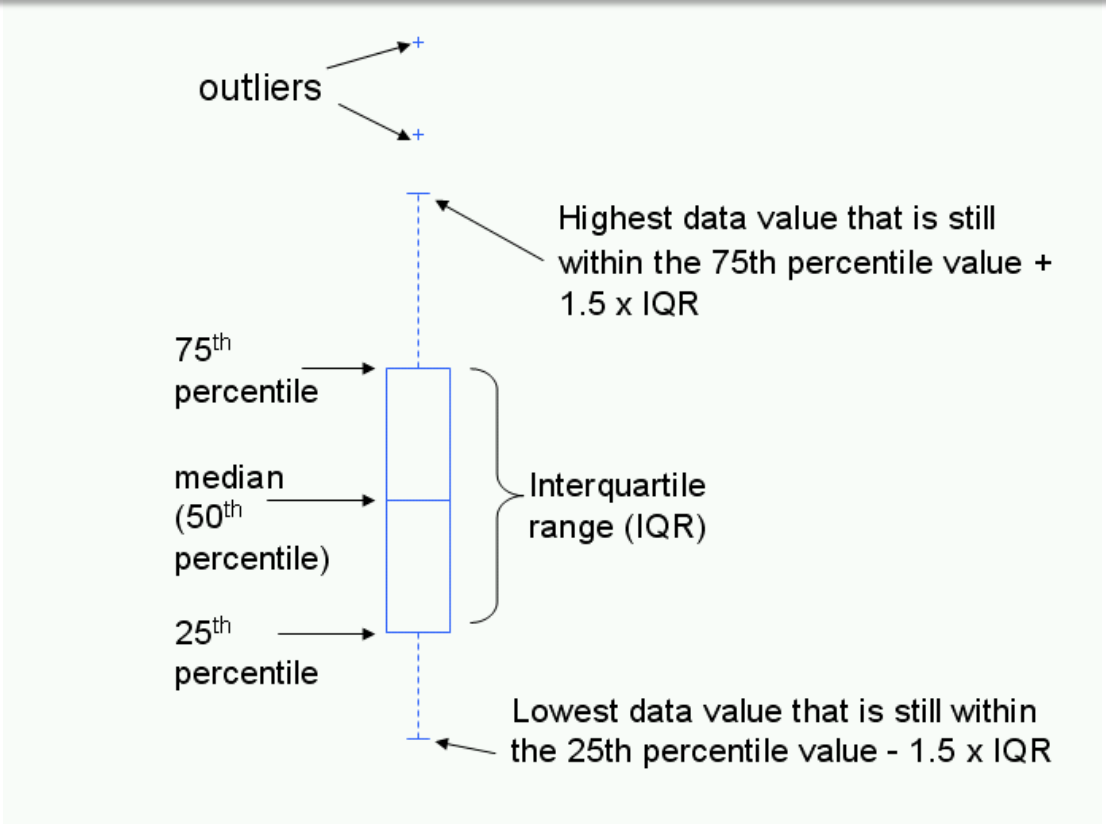


Description of arguments and possible values are described in detail in the “R Documentation“

```
?plot                            # Show documentation for plot{graphics}
```

Graphics in R
boxplot

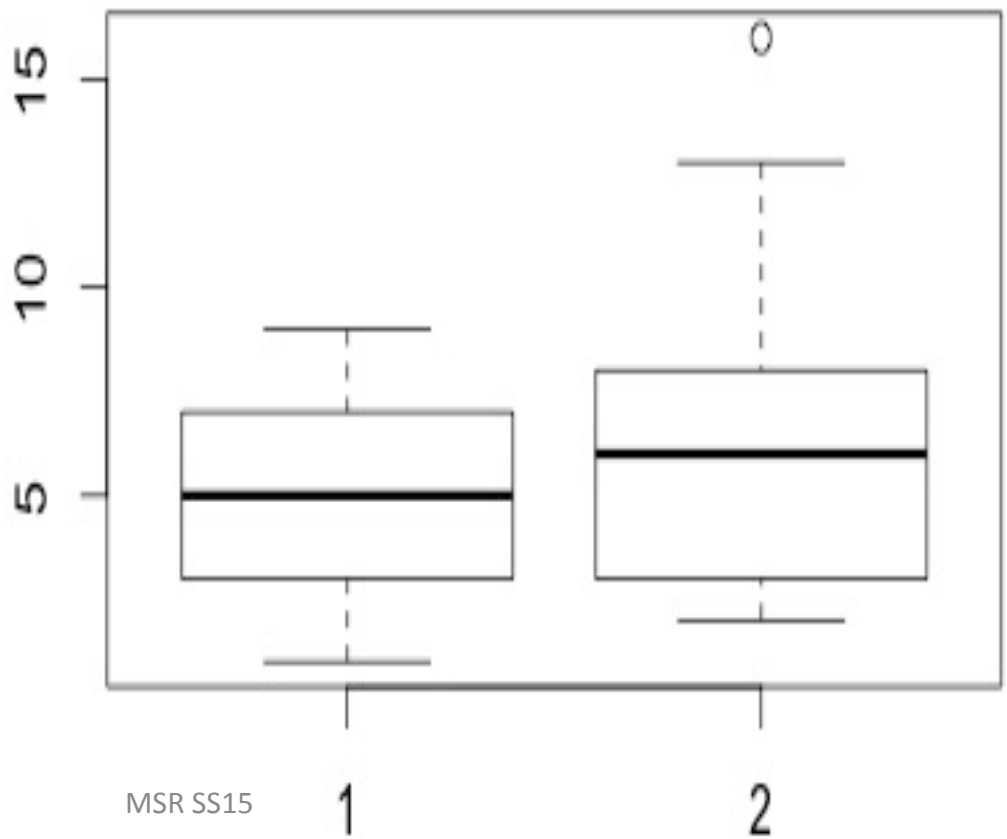
A graphic representation of a distribution, which mark the maximum and minimum values, the median and first and third quartiles. [2]



http://www.bom.gov.au/water/ssf/faq/boxplot.png

Graphics in R
boxplot

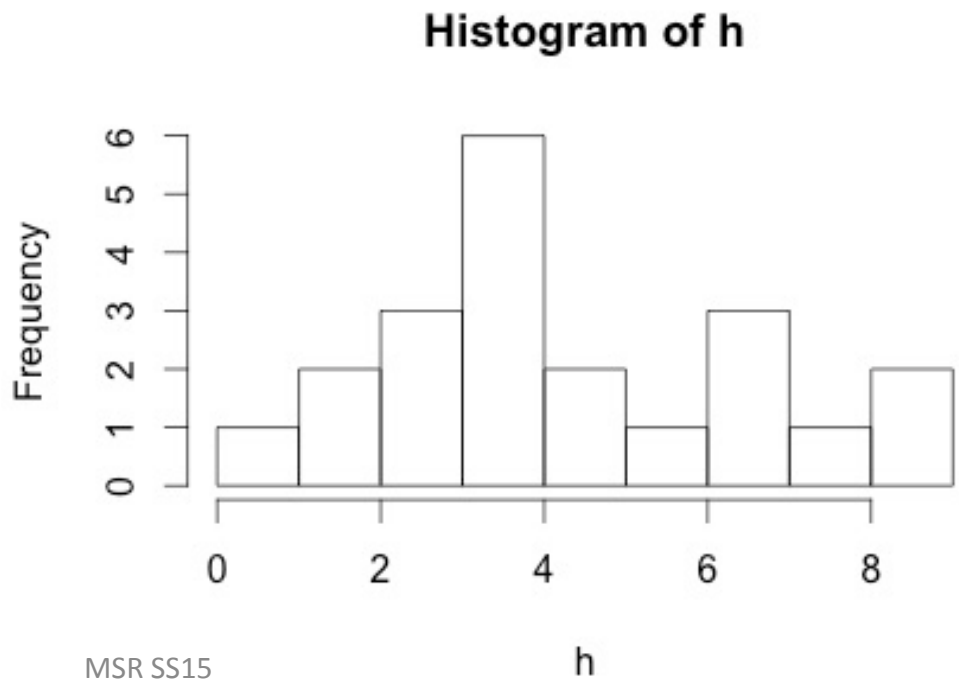
```
vx <- c(1:9)
vy <- c(6,13,8,3,5,2,7,16,3)
boxplot(vx,vy)
```



Graphics in R
histogram

A statistical graph that represents the frequency of values of a quantity. [3]

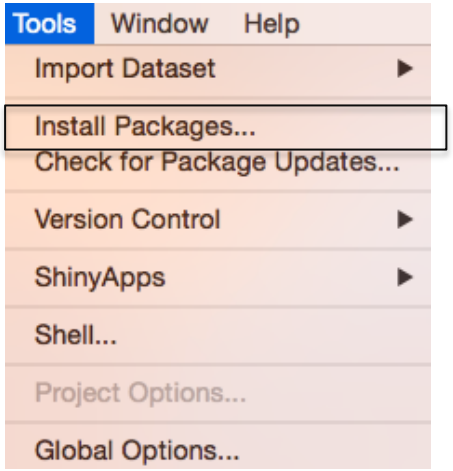
```
h <- c(4,3,7,3,5,7,1,4,2,4,8,9,6,4,2,7,4,9,5,3,4)
hist(h,breaks=0:9)
```

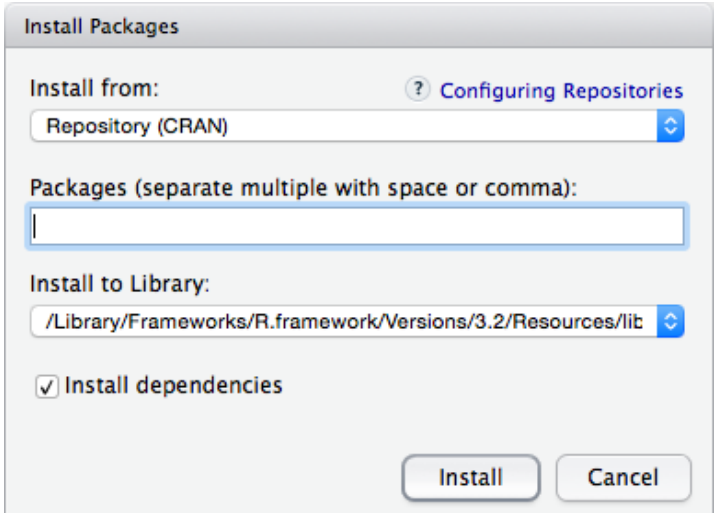


Packages in R

- library() #shows you all installed packages
- library(X) #load the package X

- Install new packages:





For more information about R:

http://cran.r-project.org/manuals.html

BASIC STATISTICS

Basic Statistics

- Number of items
- Frequnecy of values
- Mean
- Median
- Min/Max
- Quantiles
- Variance

Number of items

For a given distributen $X = [x_1,x_2,\ldots,x_n]$ is *n* the number of items (length)

Example:

X = [3,1,5,5,3,1,5,1,1,4,6,5,4,2,1]

→ n = 15

R:

length(X)

Frequnecy of values

In statistics the **frequency** (or **absolute frequency**) of an event is the number of times the event occurred in an experiment or study. These frequencies are often graphically represented in histograms. [4]

Example:

X = [3,1,5,5,3,1,5,1,1,4,6,5,4,2,1]

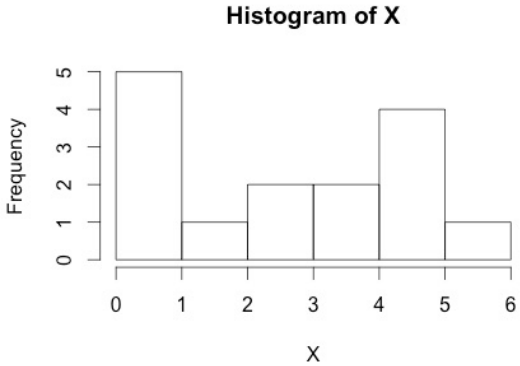
→ [1] occurs 5 times [2] occurs 1 times

[3] occurs 2 times [4] occurs 2 times

[5] occurs 4 times [6] occurs 1 times

R:

hist(X,breaks=0:6)



Mean

The *arithmetic mean* is the sum of the sampled values divided by the number of items in the sample. [5]

$$\bar{x} = \frac{x_1 + x_2 + \cdots + x_n}{n}$$

Example:

X = [3,1,5,5,3,1,5,1,1,4,6,5,4,2,1]

→ The mean is (3+1+5+5+...+4+2+1)/15 ≈ **3.133**

R:

mean(X)

summary(X)

Median

The Median is the middle number in a given sequence of numbers, taken as the mean of the two middle numbers when the sequence has an even number of items. [6]

Example:

X = [3,1,5,5,3,1,5,1,1,4,6,5,4,2,1]

X = [1,1,1,1,1,2,3,3,4,4,5,5,5,5,6]

→ The median is 3

Y = [2,4,5,8]

→ The median is 4.5

R:

median(X)

summary(X)

boxplot(X)

min/max

(verb)

Usually used in the context of roleplaying games, to min/max refers to the act of designing a character in such a way that one minimizes its weaknesses and maximizes its strengths.

Now that I know more about how the game works, I'm going to min/max my next character so that it's more effective.

http://de.urbandictionary.com/define.php?term=min%2Fmax

Min & Max

The minimum is the lowest and the maximum the highest value of a distribution

Example:

X = [3,1,5,5,3,1,5,1,1,4,6,5,4,2,1]

X = [1,1,1,1,1,2,3,3,4,4,5,5,5,5,6]

min max

R:

min(X)

max(X)

range(X)

summary(X)

boxplot(X)

Quantiles

- Important quantiles are:

0% (min), 25%, 50% (median), 75%, 100% (max)

- 9 different quantile algorithms
- Default in R: type = 7
- See ?quantile

```
> X1 =c(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16)
> summary(X1)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
   1.00    4.75    8.50    8.50   12.25   16.00 
> quantile(X1,type=1)
  0%  25%  50%  75% 100% 
   1    4    8   12   16 
> quantile(X1,type=2)
  0%  25%  50%  75% 100% 
 1.0  4.5  8.5 12.5 16.0 
> quantile(X1,type=3)
  0%  25%  50%  75% 100% 
   1    4    8   12   16 
> quantile(X1,type=4)
  0%  25%  50%  75% 100% 
   1    4    8   12   16 
> quantile(X1,type=5)
  0%  25%  50%  75% 100% 
 1.0  4.5  8.5 12.5 16.0 
> quantile(X1,type=6)
   0%   25%   50%   75%  100% 
 1.00  4.25  8.50 12.75 16.00 
> quantile(X1,type=7)
   0%   25%   50%   75%  100% 
 1.00  4.75  8.50 12.25 16.00 
> quantile(X1,type=8)
       0%       25%       50%       75%      100% 
 1.000000  4.416667  8.500000 12.583333 16.000000 
> quantile(X1,type=9)
     0%     25%     50%     75%    100% 
 1.0000  4.4375  8.5000 12.5625 16.0000 
```

Variance

The variance is a numerical measure of how the data values is dispersed around the mean.

population variance is defined in terms of the population mean μ and population size *N:*

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2$$

(σ is called standard deviation)

Example:

X = [3,1,5,5,3,1,5,1,1,4,6,5,4,2,1]

→ N = 15, $\mu \approx 3.133$ → The variance is ≈ 3.409524

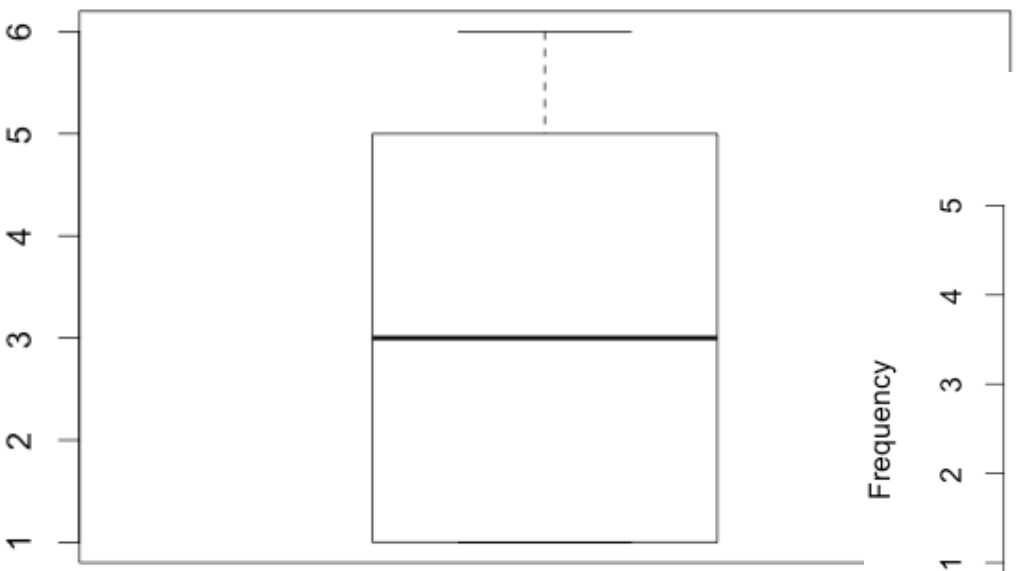
R:

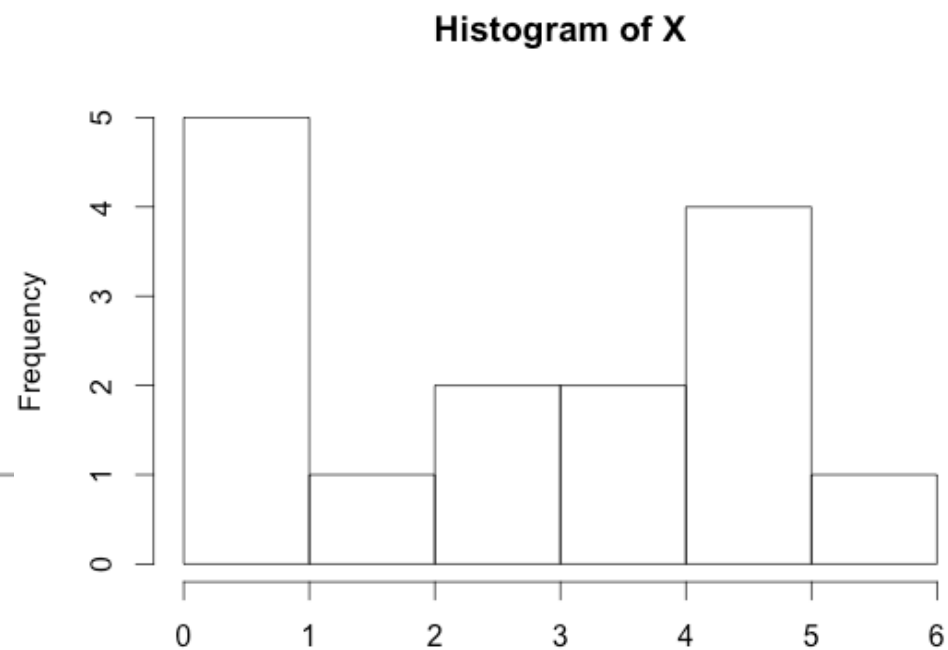
var(X)

```
> X = c(3,1,5,5,3,1,5,1,1,4,6,5,4,2,1)
```

```
> length(X)
[1] 15
```

```
> summary(X)
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max. 
  1.000   1.000   3.000   3.133   5.000   6.000 
```






```
> var(X)
[1] 3.409524
```

ADVANCED STATISTICS

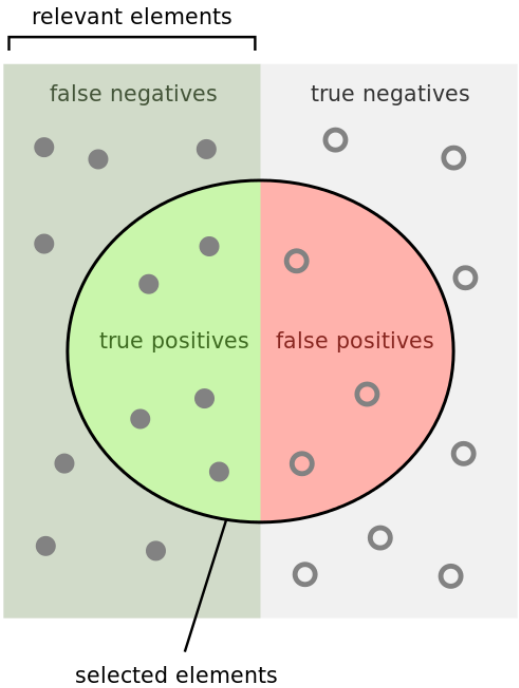
Precision, Recall, F-Measure and Accuracy

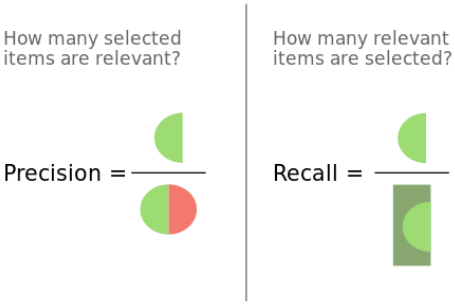
$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

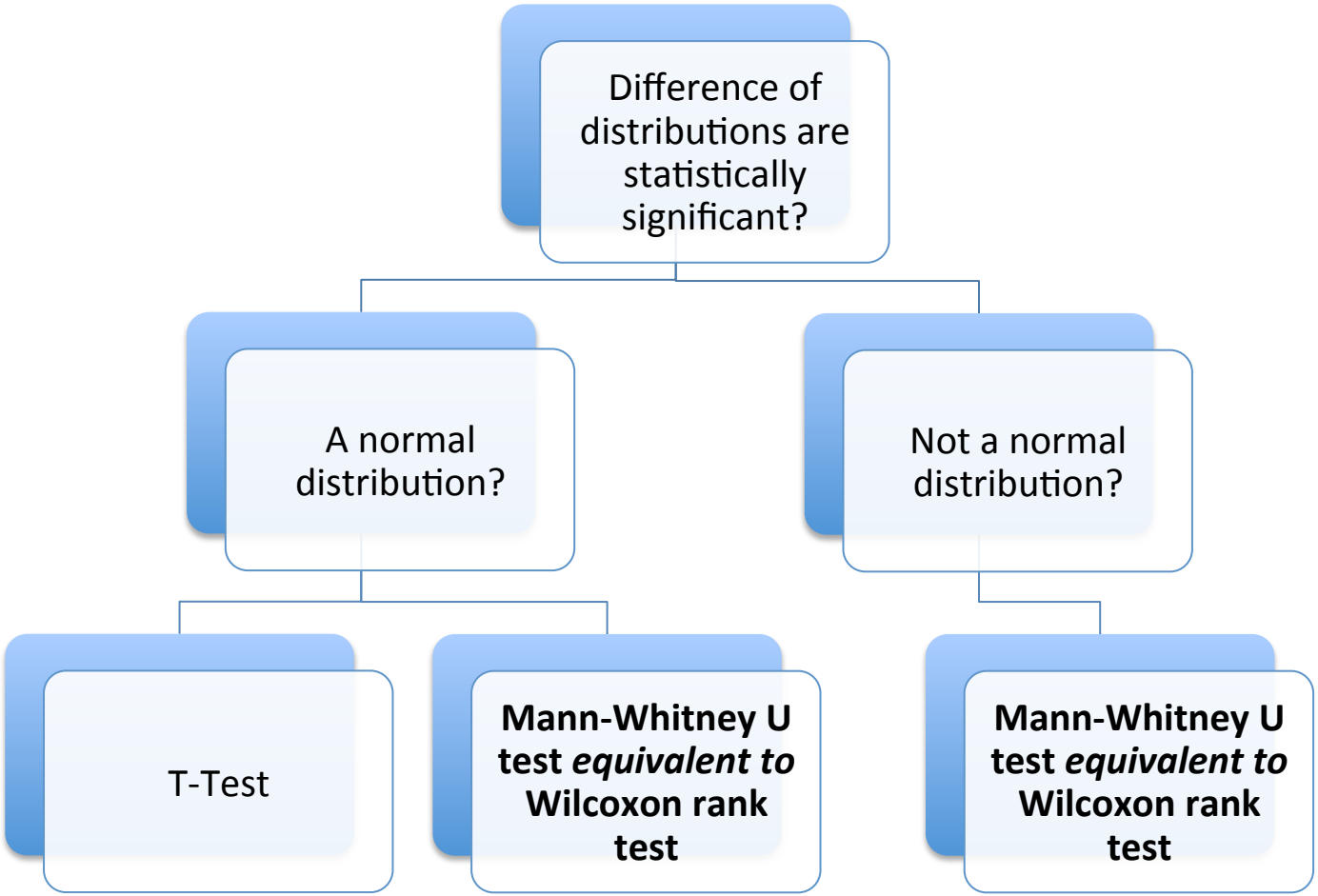
$$F = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn}$$





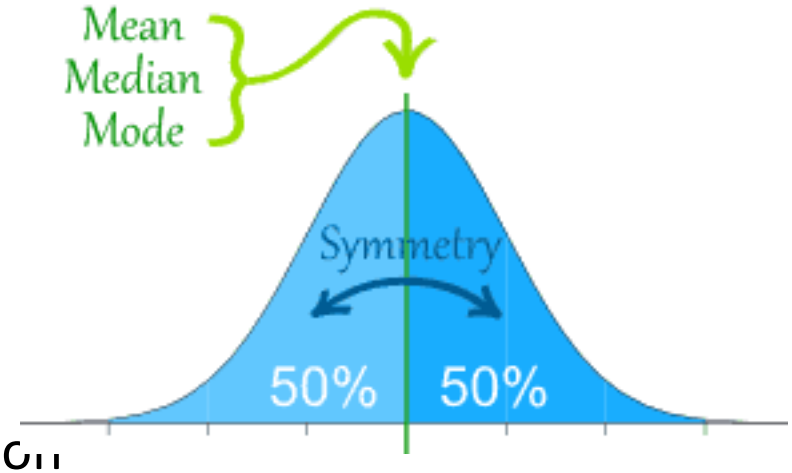
Which method?



Normal distribution

We say the data is "normally distributed“, if

- Mean = median = mode
- Symmetry about the center
- 50% of values less than the mean and 50% greater than the mean



Many things follow a Normal Distribution

- Heights of people
- Size of things produced by machines
- Errors in measurements
- Blood pressure
- Marks on a test

R

- rnorm(100) → Normal distribution with 100 elements

Normal distribution

- **Normal distribution tests** assess the likelihood that the given data set $\{x_1, \ldots, x_n\}$ comes from a normal distribution.
- There exist over 40 tests.

Frequently uesd tests:

- **"Visual" tests** are more intuitively appealing but subjective at the same time, as they rely on informal human judgement to accept or reject the null hypothesis.
  - **Q-Q plot**
  - **P-P plot**
  - **Shapiro-Wilk test**
  - **Normal probability plot**
- **Moment tests**:
  - **D'Agostino's K-squared test**
  - **Jarque–Bera test**
- **Empirical distribution function tests**:
  - **Kolmogorov–Smirnov test**
  - **Lilliefors test** (an adaptation of the Kolmogorov–Smirnov test)
  - **Anderson–Darling test**

In R (one example):

- Shapiro-Wilk test: shapiro.test(X)
  If (p-value <= 0.05) then (non-normal distribution) else (normal distribution)

Wilcoxon-Mann-Whitney test

- Also called *Mann-Whitney U test, Mann-Whitney-Wilcoxon (MWW), Wilcoxon rank-sum test (WRS)*

- **Nonparametric** test of the null hypothesis

- greater efficiency than *t*-test on non-normal distributions

- nearly as efficient as the *t*-test on normal distributions

In R:

- wilcox.test(x,...)

Wilcoxon-Mann-Whitney test

- Given are two distributions X,Y
- Define the null hypothesis H_0 and the alternative hypothesis H_a
- H_a should be what you want to show
- if (p-value < 0.05)
  - then (reject H_0 and accept H_a)
  - else (accept H_0 and reject H_a)

Cliff's delta

- use it for ordinal data
- it is a measure of how often one the values in one distribution are larger than the values in a second distribution.
- it does not require any assumptions about the shape or spread of the two distributions.

$$d = \frac{\#(x_i > x_j) - \#(x_i < x_j)}{mn}$$

- |d|<0.2 "negligible", |d|<0.5 "small", |d|<0.8 "medium", otherwise "large"

In R:

- Cliff.delta(treatment, control) (needs library(effsize)

Correlation

Correlation is a measure of the relation between two or more variables

- Correlation coefficients r can range from -1.00 to +1.00.
- -1.00 represents a perfect *negative* correlation
- +1.00 represents a perfect *positive* correlation
- 0.00 represents a lack of correlation.

- **Pearson r correlation** is used to measure the degree of the relationship between linear related variables
- **Kendall rank correlation** is used to measure the strength of dependence between two variables
- **Spearman rank correlation** is used to measure the degree of association between two variables

In R:

- Cor(X,Y,method="pearson") method can be "spearman", "kendall"
- Cor.test(X,Y,method="p|s|k")

Correlation

Examples

- r = 0.8

→positive correlation if p-value<=0.05

- r = 0.0005

→no correlation

- r = -0.2

→ if p-value<0.05 then it tends to negative correlation

EXAMPLES
HOW ARE USED STATISTICS IN MSR-FIELD?

Characterizing and Predicting Blocking Bugs in Open Source Projects

Context:

- They propose a model to predict blocking-bugs
- *Blocking bugs* are software bugs that prevent other bugs from being fixed.

Motivation:

- These blocking bugs may increase maintenance costs, reduce overall quality and delay the release of the software systems.

Artifacts:

- They study six open source projects (Chromium, Eclipse, FreeDesktop, Mozilla, Netbeans, Open-Office)

Research Question

- RQ1: Can we build highly accurate models to predict whether a new bug will be a blocking bug?
- RQ2: Which factors are the best indicators of blocking bugs?

http://dl.acm.org/citation.cfm?doid=2597073.2597099

Characterizing and Predicting Blocking Bugs in Open Source Projects

- They use „Prediction models“ to identify whether a bug will be a blocking bug or not
  - Decision Tree Classifier
  - Naive Bayes Classifier
  - K-Nearest Neighbor Classifier
  - Random Forest Classifier
  - Zero-R Classifier

http://dl.acm.org/citation.cfm?doid=2597073.2597099

Characterizing and Predicting Blocking Bugs in Open Source Projects

Table 3: Median fixing time in days

Project	Blocking	Non-blocking
Chromium	48	16
Eclipse	146	51
FreeDesktop	78	37
Mozilla	100	42
Netbeans	225	134
Open-Office	142	74

Median

Table 4: Wilcoxon-Test for blocking and non-blocking bugs

Project	$t_{blocking} - t_{nonblocking}$
Chromium	15.9 ***
Eclipse	52.7 ***
FreeDesktop	19.2 ***
Mozilla	37.0 ***
Netbeans	16.8 ***
Open-Office	40.1 ***

(***) $p < 0.001$

The median time to address a blocking bug 15-40 days longer than the median time it takes to address a non-blocking bug.

Table 5: Median time to identify a blocking bug in days

Project	Time to identify a blocking bug
Chromium	8
Eclipse	17
FreeDesktop	9
Mozilla	18
NetBeans	5
OpenOffice	3

The median time it takes to identify a blocking bug is 3-18 days

To see if there is a significant difference between the blocking and non-blocking bugs, they performed an **Wilcoxon-test** for the hypothesis $H_a : t_{blocking} > t_{nonblocking}$. Table 4 shows the fixing-time difference reported by the test. The ***p-values* were significant for all projects (*p-value* < 0.001)**, meaning that **the fixing-time for blocking bugs is statistically significantly longer than the fixing-time for non-blocking bugs**.

http://dl.acm.org/citation.cfm?doid=2597073.2597099

Characterizing and Predicting Blocking Bugs in Open Source Projects

Table 9: Predictions different algorithms

Project	Classif.	Precision	Recall	F-measure	Acc.
Chromium	Zero-R	NA	0%	0%	**97.6%**
	Naive Bayes	10.0%	54.2%	16.9%	81.5%
	kNN	9.0%	47.1%	15.1%	81.5%
	Rand. Forest	**18.6%**	29.6%	**22.8%**	93.0%
	C4.5	9.1%	**49.9%**	15.3%	80.7%
Eclipse	Zero-R	NA	0%	0%	**97.2%**
	Naive Bayes	8.8%	**66.4%**	15.5%	79.7%
	kNN	8.1%	53.0%	14.0%	81.8%
	Rand. Forest	**16.5%**	24.0%	**19.5%**	94.5%
	C4.5	9.2%	47.0%	15.4%	85.5%
FreeDesktop	Zero-R	NA	0%	0%	**91.1%**
	Naive Bayes	18.7%	**74.3%**	29.9%	69.0%
	kNN	19.4%	72.6%	30.6%	70.7%
	Rand. Forest	**27.8%**	60.2%	**37.9%**	82.4%
	C4.5	20.4%	73.6%	31.9%	72.0%
Mozilla	Zero-R	NA	0%	0%	**87.4%**
	Naive Bayes	29.5%	68.0%	41.1%	75.6%
	kNN	23.3%	69.3%	34.9%	67.5%
	Rand. Forest	**36.1%**	53.6%	**43.2%**	82.3%
	C4.5	29.0%	**76.7%**	42.1%	73.6%
NetBeans	Zero-R	NA	0%	0%	**96.8%**
	Naive Bayes	9.9%	**73.3%**	17.3%	77.1%
	kNN	11.4%	59.3%	19.1%	84.2%
	Rand. Forest	**26.3%**	37.6%	**30.9%**	94.7%
	C4.5	12.8%	59.3%	21.1%	86.0%
OpenOffice	Zero-R	NA	0%	0%	**96.9%**
	Naive Bayes	12.9%	**78.1%**	22.1%	83.3%
	kNN	14.2%	66.1%	23.4%	87.0%
	Rand. Forest	**32.9%**	46.7%	**38.6%**	95.5%
	C4.5	15.9%	65.9%	25.6%	88.4%

		True class	
		Blocking	Non-blocking
Classified as	Blocking	TP	FP
	Non-blocking	FN	TN

1. Precision: The ratio of correctly classified blocking bugs over all the bugs classified as blocking. It is calculated as $Pr = \frac{TP}{TP+FP}$.
2. Recall: The ratio of correctly classified blocking bugs over all of the actually blocking bugs. It is calculated as $Re = \frac{TP}{TP+FN}$.
3. F-measure: Measures the weighted harmonic mean of the precision and recall. It is calculated as $F\text{-}measure = \frac{2*Pr*Re}{Pr+Re}$.
4. Accuracy: The ratio between the number of correctly classified bugs (both the blocking and the non-blocking) over the total number of bugs. It is calculated as $Acc = \frac{TP+TN}{TP+FP+TN+FN}$.

http://dl.acm.org/citation.cfm?doid=2597073.2597099

An Empirical Study of Dormant Bugs

Context

- They study dormant bugs against non-dormant bugs
- Dormant bugs are bugs which are introduced in a version of the software system and are found and reported much later (e.g. after three versions or after one year)

Motivation:

- Since some bugs may not be reported till much later, us-ing only non-dormant bugs to evaluate the overall quality of a version of a system may be inaccurate. If many bugs are reported in a version, but these bugs were introduced in previous version, then the quality of that version may be misjudged.

Artifacts:

- 20 Open-Source Apache foundation software systems

http://dl.acm.org/citation.cfm?doid=2597073.2597108

An Empirical Study of Dormant Bugs

Research Question:

RQ1: How quickly are dormant bugs fixed?

Dormant bugs are fixed faster than non-dormant bugs.

Dormant bug fixes are more complex in terms of lines touched and files modified, when compared to non-dormant bugs. However, the difference between number of files modified is small (median dormant bug fixes modify 9 more lines of code and same number of files than non-dormant bugs).

RQ2: What is the size of a dormant bug fix?

RQ3: Who fixes dormant bugs?

Dormant bugs are fixed by more experienced developers.

Our manual analysis shows that dormant bugs are caused more often by control flow and corner case problems, when compared to non-dormant bugs. Since such problems are harder to debug, dormant bugs may be assigned to more experienced developers. The results from our manual study can help researchers design better dormant bug detection techniques, by focussing on specific sub-categories of root causes.

RQ4: What are the root causes of dormant bug fixes?

http://dl.acm.org/citation.cfm?doid=2597073.2597108

An Empirical Study of Dormant Bugs

- They used the **Wilcoxon rank-sum test** to compare the *fix time of the dormant* and *non-dormant bugs*
- The dataset is highly skewed → no normal distribution → t-Test not possible
- They want to show that dormant bugs have a shorter fix time → alternative hypothesis is set to less
- p-value is << 0.001.

  →They accept the alternative hypothesis.
  This indicates that the fix time for dormant bugs is statistically significantly smaller than that of non-dormant bugs.

http://dl.acm.org/citation.cfm?doid=2597073.2597108

An Empirical Study of Dormant Bugs

- Correlation between *dormant time* and *fix time*
- Correlation value r = 0.073

→*dormant time* does not have any relationship with *fix time*

http://dl.acm.org/citation.cfm?doid=2597073.2597108

Mining StackOverflow to Turn the IDE into a Self-Confident Programming Prompter

Context

- Analysis of StackOverflow

Contribution

- a novel ranking model that evaluates the relevance of a Stack Overflow discussion, given a code context in the IDE, by considering code, conceptual and community aspects
- An Eclipse plug-in named PROMPTER

http://dl.acm.org/citation.cfm?doid=2597073.2597077

Mining StackOverflow to Turn the IDE into a Self-Confident Programming Prompter

Generate a query with source code →

Table 1: PROMPTER Ranking Model: Best Configuration.

Index	Weight	Index	Weight
Textual Similarity	0.32	Question Score	0.07
Code Similarity	0.00	Accepted Answer Score	0.00
API Types Similarity	0.00	User Reputation	0.13
API Methods Similarity	0.30	Tags Similarity	0.18

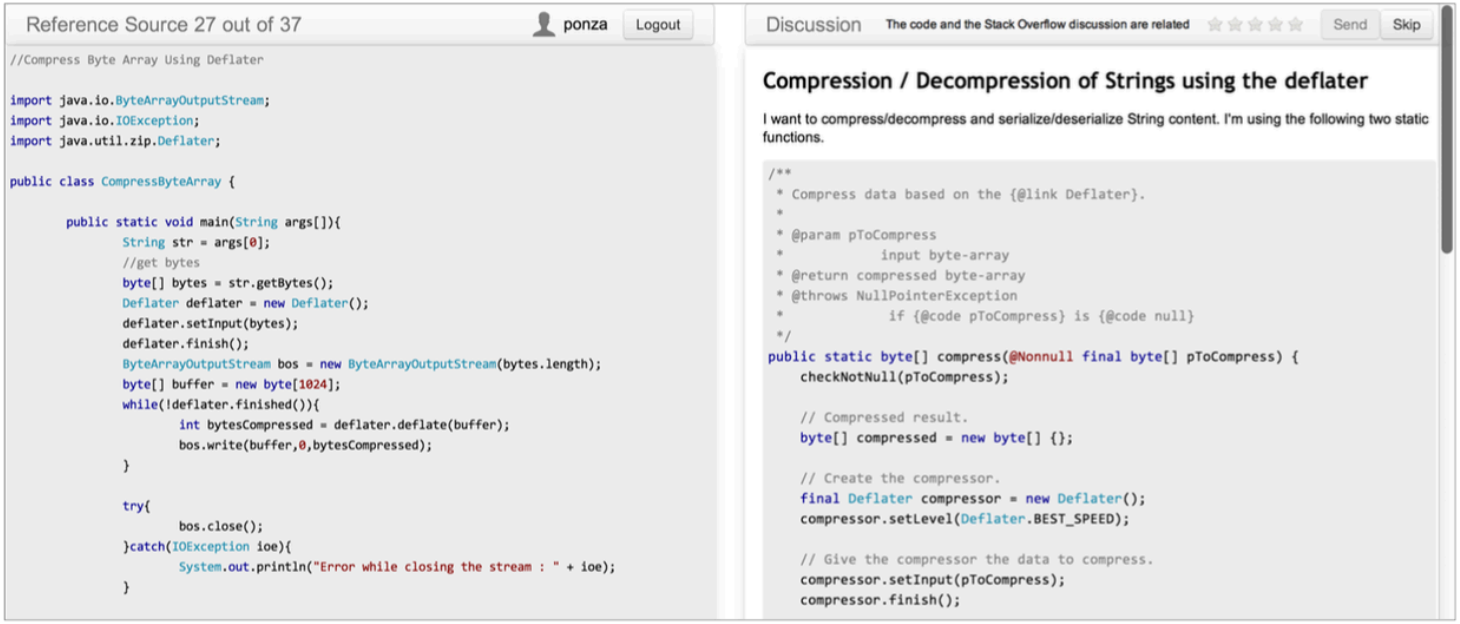


Figure 3: An Example Question from Our Questionnaire.

http://dl.acm.org/citation.cfm?doid=2597073.2597077

Mining StackOverflow to Turn the IDE into a Self-Confident Programming Prompter

- Evaluate PROMPTER with developers
  - Participants have similar experience
  - Participants perform Maintnance Task (MT) and Development Task (DT)
  - Participants work with PROMPTER(P) and without PROMPTER(NP)

http://dl.acm.org/citation.cfm?doid=2597073.2597077

Mining StackOverflow to Turn the IDE into a Self-Confident Programming Prompter

4.3 Quantitative Analysis of the Results

Look at "Overall":

- The P median is 68% (mean 70%) against the 40% median (mean 46%) of NP.

→ In other words, Prompter allowed participants to achieve a median additional completeness of 28% (mean of 24%).

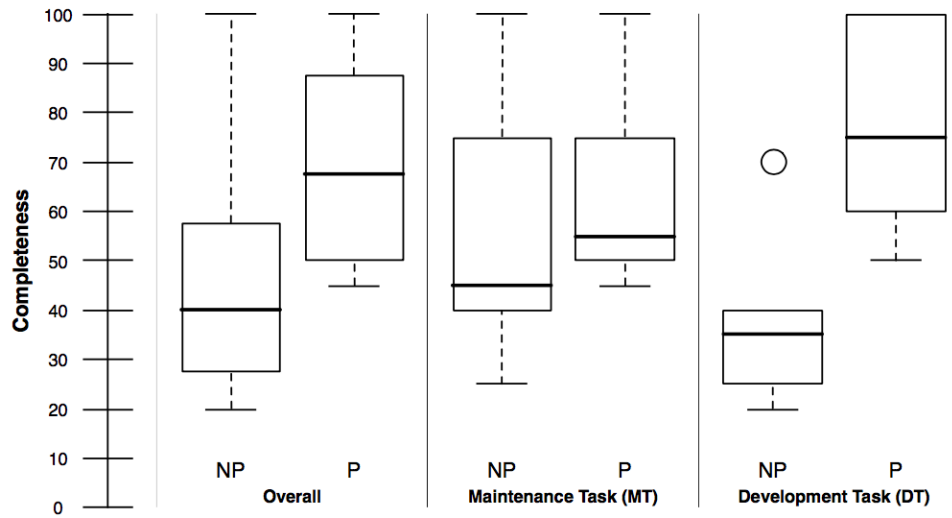


Figure 5: Boxplots of Completeness achieved by Participants with (P) and without (NP) PROMPTER.

http://dl.acm.org/citation.cfm?doid=2597073.2597077

Mining StackOverflow to Turn the IDE into a Self-Confident Programming Prompter

- Statical significance of the difference?

- Use Cliff's delta (d) to assess the magnitude of such a difference
- Use Wilcoxon-Mann-Whitney test
  - Overall:
    - *p*-value <0.01 → statistically significant
    - d = 0.65 → large effect

  - MT
    - not significant (*p*-value=0.23)
    - d = 0.38 medium effect,
  - DT
    - *p*-value=0.03 → statistically significant
    - d = 0.88 large effect

http://dl.acm.org/citation.cfm?doid=2597073.2597077

literature

[1] http://www.r-project.org/

[2] http://dictionary.reference.com/browse/box plot

[3] http://dictionary.reference.com/browse/histogram

[4] http://en.wikipedia.org/wiki/Frequency_(statistics)

[5] http://en.wikipedia.org/wiki/Mean

[6] http://dictionary.reference.com/browse/median

[7] http://www.r-tutor.com/elementary-statistics/non-parametric-methods/mann-whitney-wilcoxon-test

[8] http://en.wikipedia.org/wiki/Precision_and_recall

[9] H. V. Garcia, E. Shibab: Characterizing and Predicting Blocking Bugs in Open Source Projects, MSR14, http://dl.acm.org/citation.cfm?doid=2597073.2597099

[10] T. Chen, M. Nagappan, E.Shibab, A.E.Hassan: An Empirical Study of Dormant Bugs, MSR14, http://dl.acm.org/citation.cfm?doid=2597073.2597108

[11]L. Ponzanelli, G. Bavota, M.Di Penta, R. Oliveto, M. Michele Lanza: Mining StackOverflow to Turn the IDE into a Self-Confident Programming Prompter, MSR14, http://dl.acm.org/citation.cfm?doid=2597073.2597077

[12] http://www.r-project.org/

Question?