UT Dallas

Software Quality and Software Testing

Part 1 – The Big Picture (How Quality Relates to Testing)

Part 2 – Fundamental Concepts of Measurement and Data Analysis

Part 3 – Defect Containment

Part 4 – Measuring Software Structure

Part 5 – Measuring Software Complexity

Dennis J. Frailey

Retired Principal Fellow - Raytheon Company

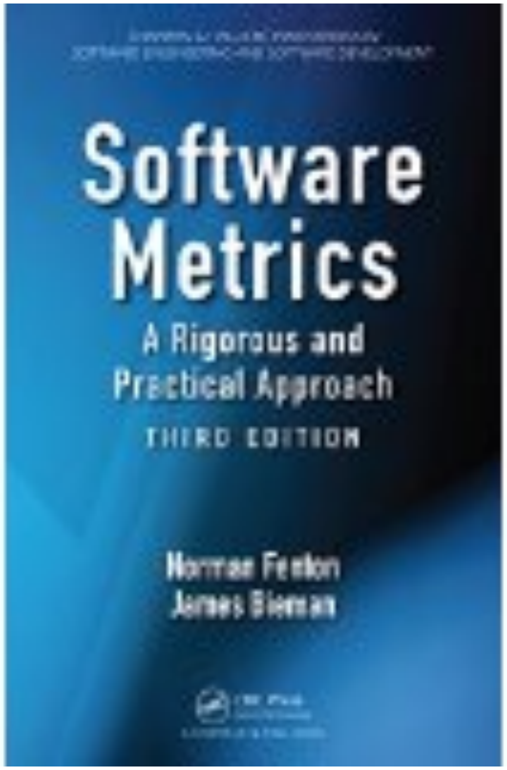
PhD Purdue, 1971, Computer Science
Assistant Professor, SMU, 1970-75
Associate Professor, SMU, 1975-77
(various titles), Texas Instruments, 1974-1997;
Raytheon Co. 1997-2010
Adjunct Associate Professor, UT Austin, 1981-86
Adjunct Professor, SMU, 1987-2017
Adjunct Professor, UT Arlington, 2014-present

Areas of specialty: software development process, software project management, software quality engineering, software metrics, compiler design, operating system design, real-time system design, computer architecture

A Recommended Book on Measurement

Some of the material covered today is taken *from this book*.

Although not a book on testing, it is a very good book on measurement and addresses several aspects of testing.



Software Metrics – A Rigorous and Practical Approach

By Norman Fenton and James Bieman

More Recommended References

***SWX – The Software Extension to the Project Management Body of Knowledge*, available from PMI (www.pmi.com) and the IEEE Computer Society (www.computer.org).**

- This is a general reference that may be important if you want to apply some of today's techniques in project management.

***SWEBOK – The Guide to the Software Engineering Body of Knowledge*, available from the IEEE Computer Society and also at www.swebok.org**

- This is another general reference that gives an overall picture of software engineering knowledge and summarizes topics that any software engineer should know about.

Part 1

The Big Picture – How Quality Relates to Testing and Other Aspects of Software Engineering

Test and Evaluation

***Evaluation*: Appraising a product through one of the following:**

- Examination, analysis, demonstration
- Testing
- or other means

***Testing*: Exercising a system to improve confidence that it satisfies requirements or to identify variations between desired and actual behavior.**

“Evaluation” is the broader term.

But What Are We Appraising?
What is “Desired Behavior”?

- Satisfies requirements
- Works correctly
- Does what I want it to do
- Does no harm
- Reliable – I can depend on it
- Easy to use
- Portable
- Easy to update and maintain
- Easy to test
- Runs efficiently / fast
- Consistent
- …

Can we test for these characteristics?

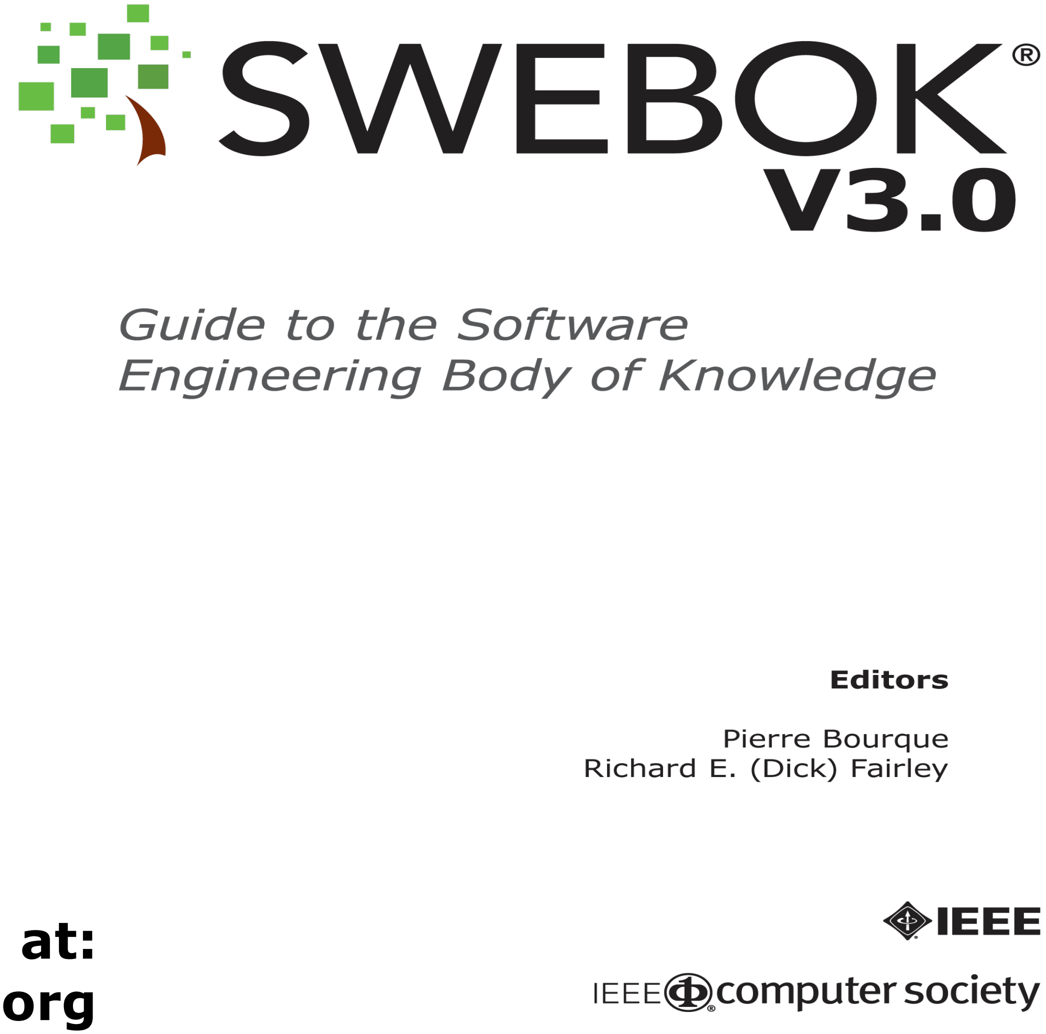
Can we measure them?

But What Are We Appraising?
What is “Desired Behavior”?

- Satisfies requirements
- Works correctly
- Does what I want it to do
- Does no harm
- Reliable – I can depend on it
- Easy to use
- Portable
- Easy to update and maintain
- Easy to test
- Runs efficiently / fast
- Consistent
- …

These are all characteristics of Software Quality

I.e., testing is one way to assess software quality.



SWEBOK®
V3.0

Guide to the Software Engineering Body of Knowledge

Editors

Pierre Bourque
Richard E. (Dick) Fairley

IEEE
IEEE computer society

Downloadable at:
www.swebok.org

SWEBOK Facts

- **3 Editions have been produced since 1998**
- **2 Editors: Pierre Bourque and Richard Fairley**
- **8 Contributing and Co-Editors**
- **15 Knowledge Areas, each with its own Editors**
  - Each aligned with related ISO and IEEE standards
- **9-person Change Control Board**
- **Over 300 reviewers (chosen due to their expertise in various aspects of software engineering)**
  - Over 1500 comments received and adjudicated on various drafts (3rd edition)
- **36 Items in Consolidated Reference List**

The 15 SWEBOK Knowledge Areas

Software Requirements

Software Design

Software Construction

Software Testing

Software Maintenance

Software Configuration Management

Software Engineering Management

Software Engineering Process

Software Engineering Models and Methods

Software Quality

Software Engineering Professional Practice

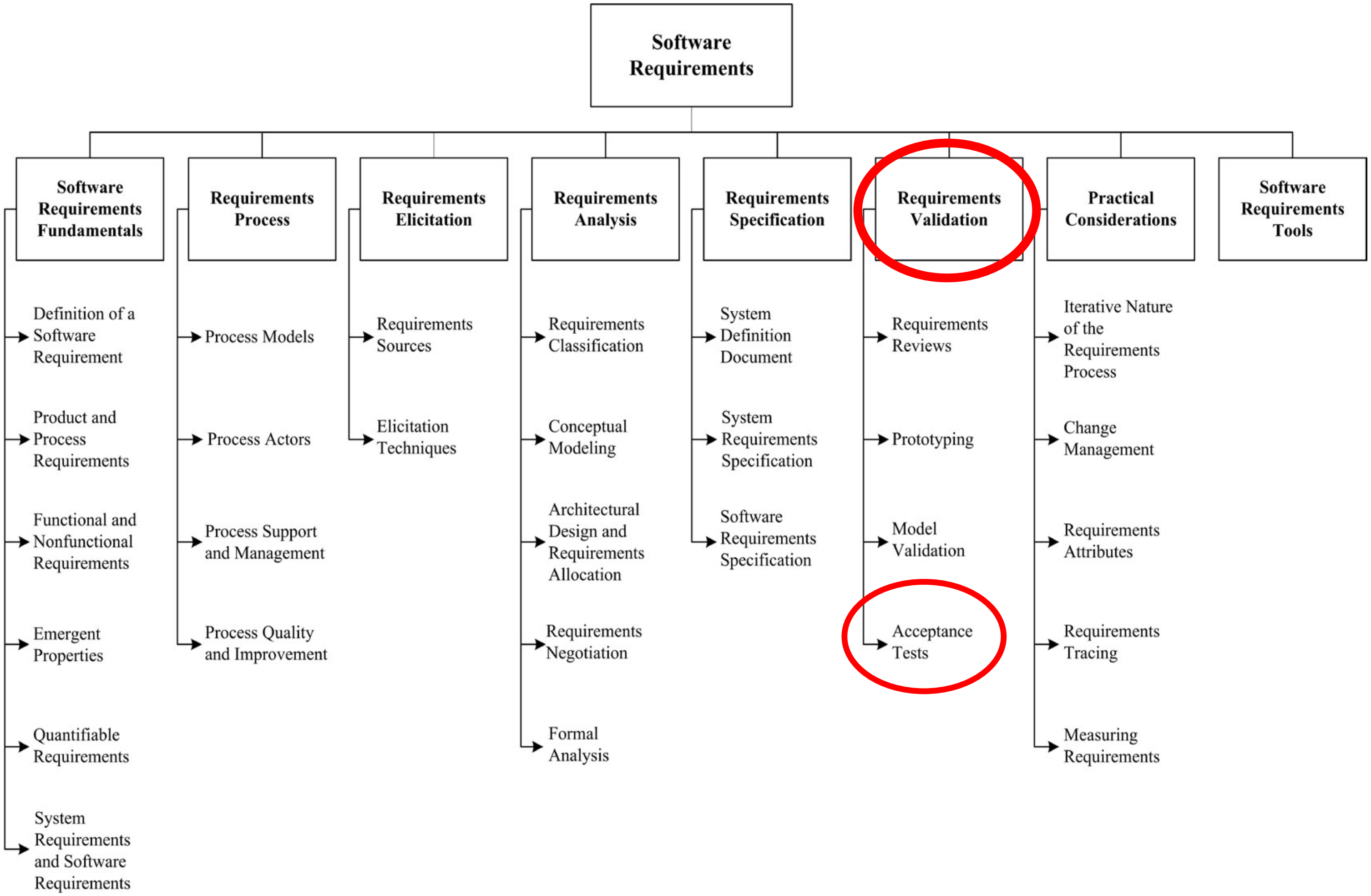
Software Engineering Economics

Computing Foundations

Mathematical Foundations

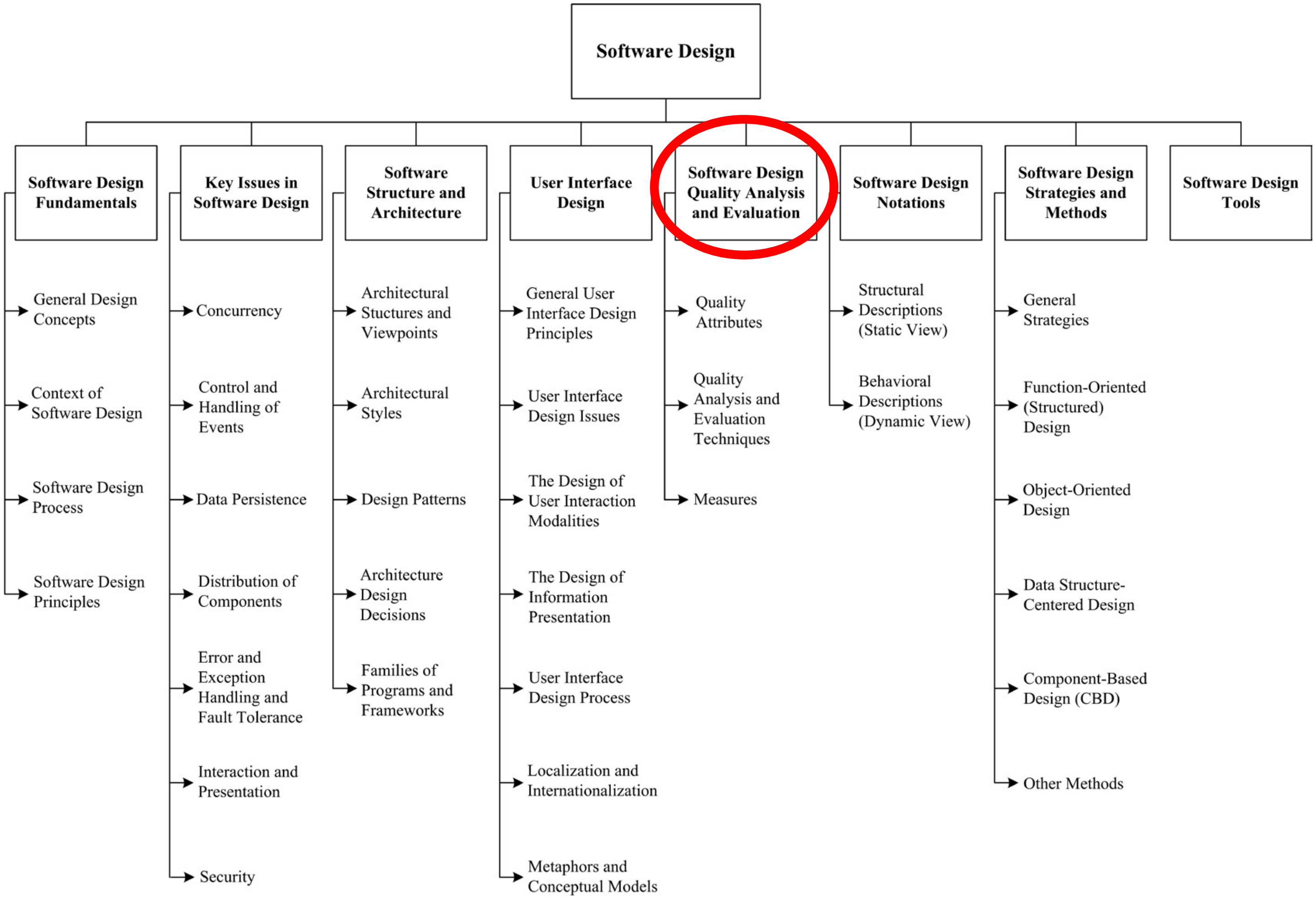
Engineering Foundations

Software Requirements



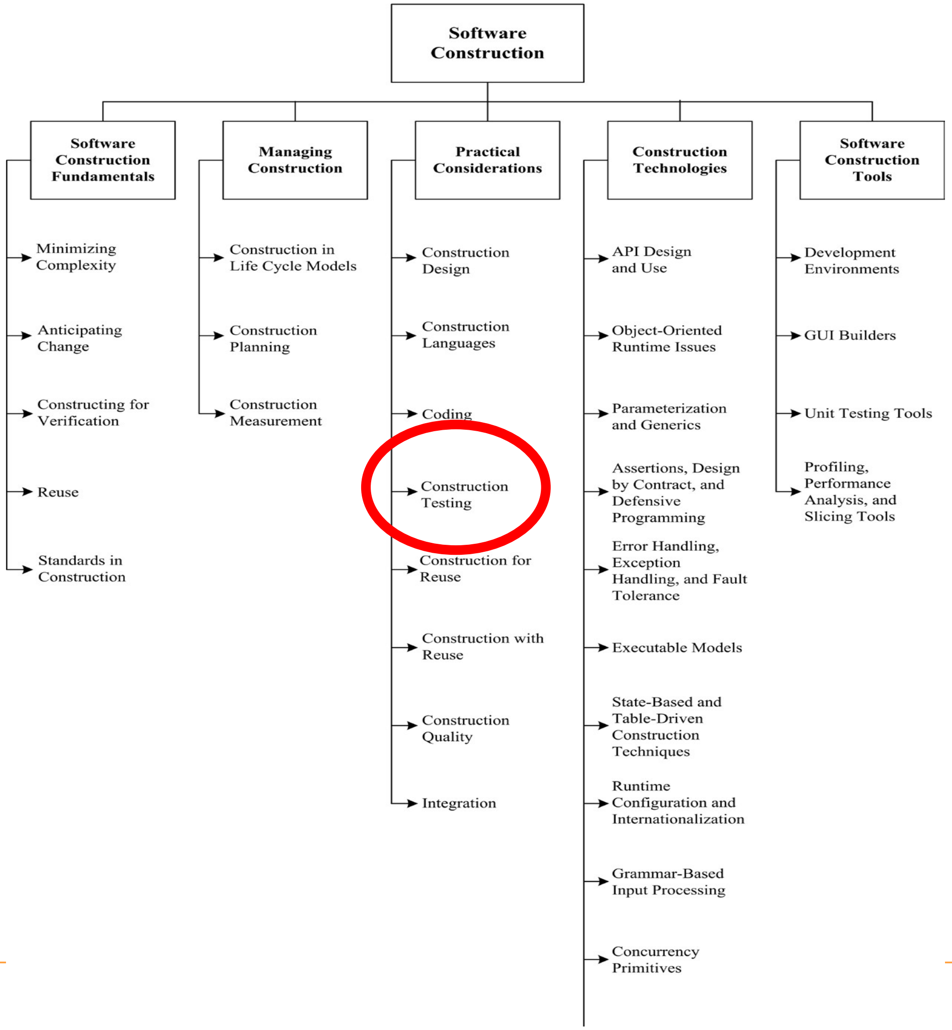
- Software Requirements
  - Software Requirements Fundamentals
    - Definition of a Software Requirement
    - Product and Process Requirements
    - Functional and Nonfunctional Requirements
    - Emergent Properties
    - Quantifiable Requirements
    - System Requirements and Software Requirements
  - Requirements Process
    - Process Models
    - Process Actors
    - Process Support and Management
    - Process Quality and Improvement
  - Requirements Elicitation
    - Requirements Sources
    - Elicitation Techniques
  - Requirements Analysis
    - Requirements Classification
    - Conceptual Modeling
    - Architectural Design and Requirements Allocation
    - Requirements Negotiation
    - Formal Analysis
  - Requirements Specification
    - System Definition Document
    - System Requirements Specification
    - Software Requirements Specification
  - Requirements Validation
    - Requirements Reviews
    - Prototyping
    - Model Validation
    - Acceptance Tests
  - Practical Considerations
    - Iterative Nature of the Requirements Process
    - Change Management
    - Requirements Attributes
    - Requirements Tracing
    - Measuring Requirements
  - Software Requirements Tools

Software Design

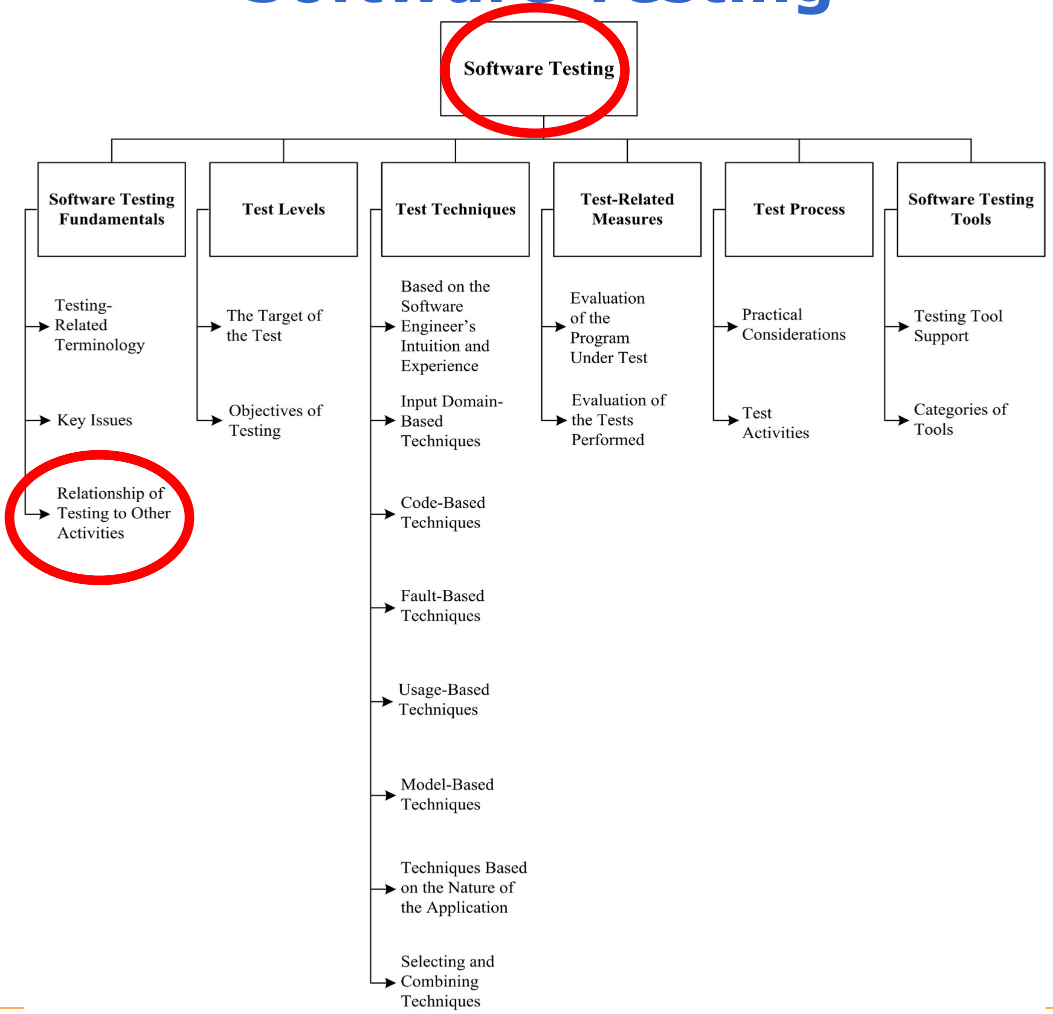


- Software Design
  - Software Design Fundamentals
    - General Design Concepts
    - Context of Software Design
    - Software Design Process
    - Software Design Principles
  - Key Issues in Software Design
    - Concurrency
    - Control and Handling of Events
    - Data Persistence
    - Distribution of Components
    - Error and Exception Handling and Fault Tolerance
    - Interaction and Presentation
    - Security
  - Software Structure and Architecture
    - Architectural Stuctures and Viewpoints
    - Architectural Styles
    - Design Patterns
    - Architecture Design Decisions
    - Families of Programs and Frameworks
  - User Interface Design
    - General User Interface Design Principles
    - User Interface Design Issues
    - The Design of User Interaction Modalities
    - The Design of Information Presentation
    - User Interface Design Process
    - Localization and Internationalization
    - Metaphors and Conceptual Models
  - Software Design Quality Analysis and Evaluation
    - Quality Attributes
    - Quality Analysis and Evaluation Techniques
    - Measures
  - Software Design Notations
    - Structural Descriptions (Static View)
    - Behavioral Descriptions (Dynamic View)
  - Software Design Strategies and Methods
    - General Strategies
    - Function-Oriented (Structured) Design
    - Object-Oriented Design
    - Data Structure-Centered Design
    - Component-Based Design (CBD)
    - Other Methods
  - Software Design Tools

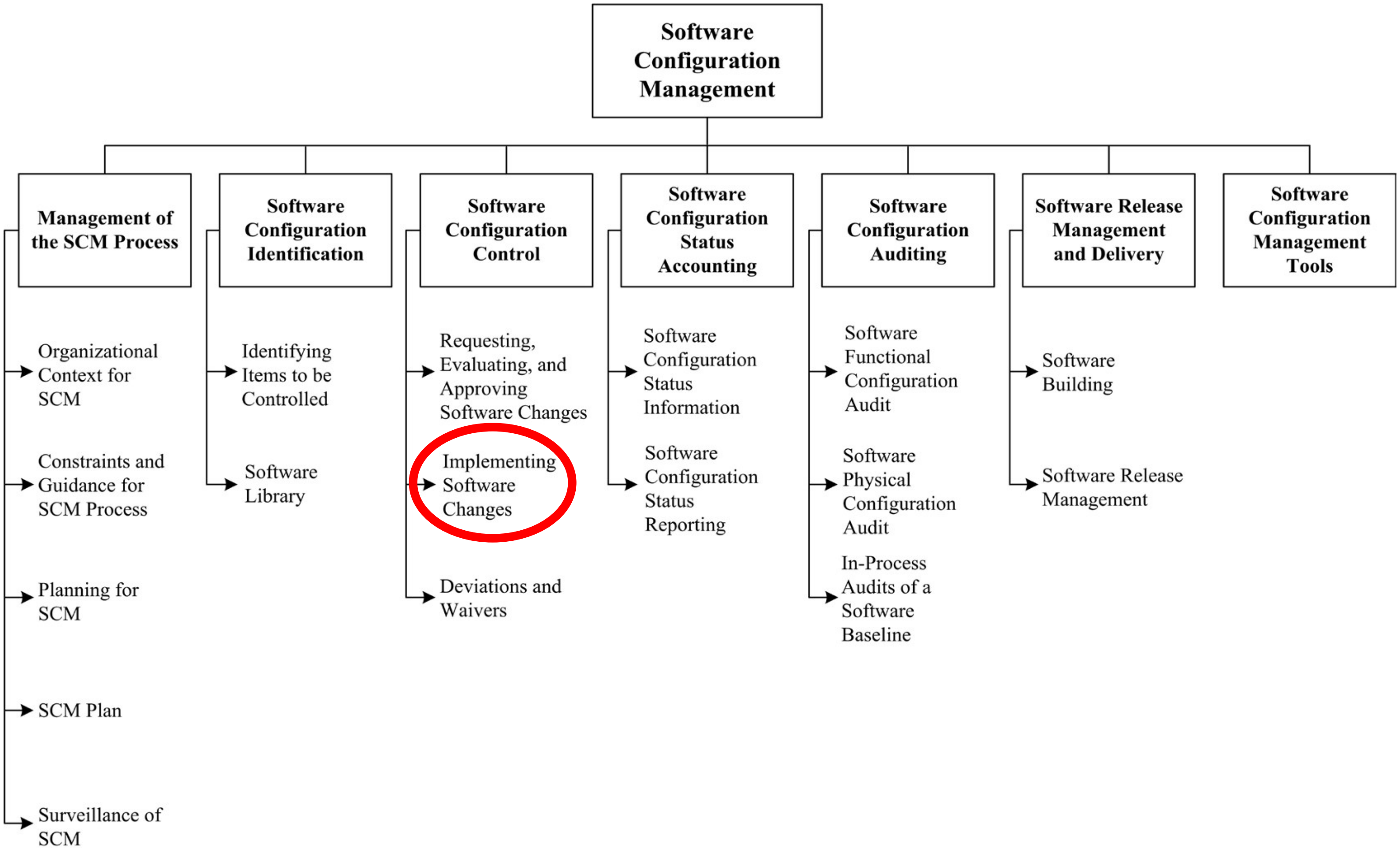
Software Construction


Software Construction
Software Construction Fundamentals
Minimizing Complexity
Anticipating Change
Constructing for Verification
Reuse
Standards in Construction
Managing Construction
Construction in Life Cycle Models
Construction Planning
Construction Measurement
Practical Considerations
Construction Design
Construction Languages
Coding
Construction Testing
Construction for Reuse
Construction with Reuse
Construction Quality
Integration
Construction Technologies
API Design and Use
Object-Oriented Runtime Issues
Parameterization and Generics
Assertions, Design by Contract, and Defensive Programming
Error Handling, Exception Handling, and Fault Tolerance
Executable Models
State-Based and Table-Driven Construction Techniques
Runtime Configuration and Internationalization
Grammar-Based Input Processing
Concurrency Primitives
Software Construction Tools
Development Environments
GUI Builders
Unit Testing Tools
Profiling, Performance Analysis, and Slicing Tools

Software Testing


Software Testing
Software Testing Fundamentals
Testing-Related Terminology
Key Issues
Relationship of Testing to Other Activities
Test Levels
The Target of the Test
Objectives of Testing
Test Techniques
Based on the Software Engineer's Intuition and Experience
Input Domain-Based Techniques
Code-Based Techniques
Fault-Based Techniques
Usage-Based Techniques
Model-Based Techniques
Techniques Based on the Nature of the Application
Selecting and Combining Techniques
Test-Related Measures
Evaluation of the Program Under Test
Evaluation of the Tests Performed
Test Process
Practical Considerations
Test Activities
Software Testing Tools
Testing Tool Support
Categories of Tools

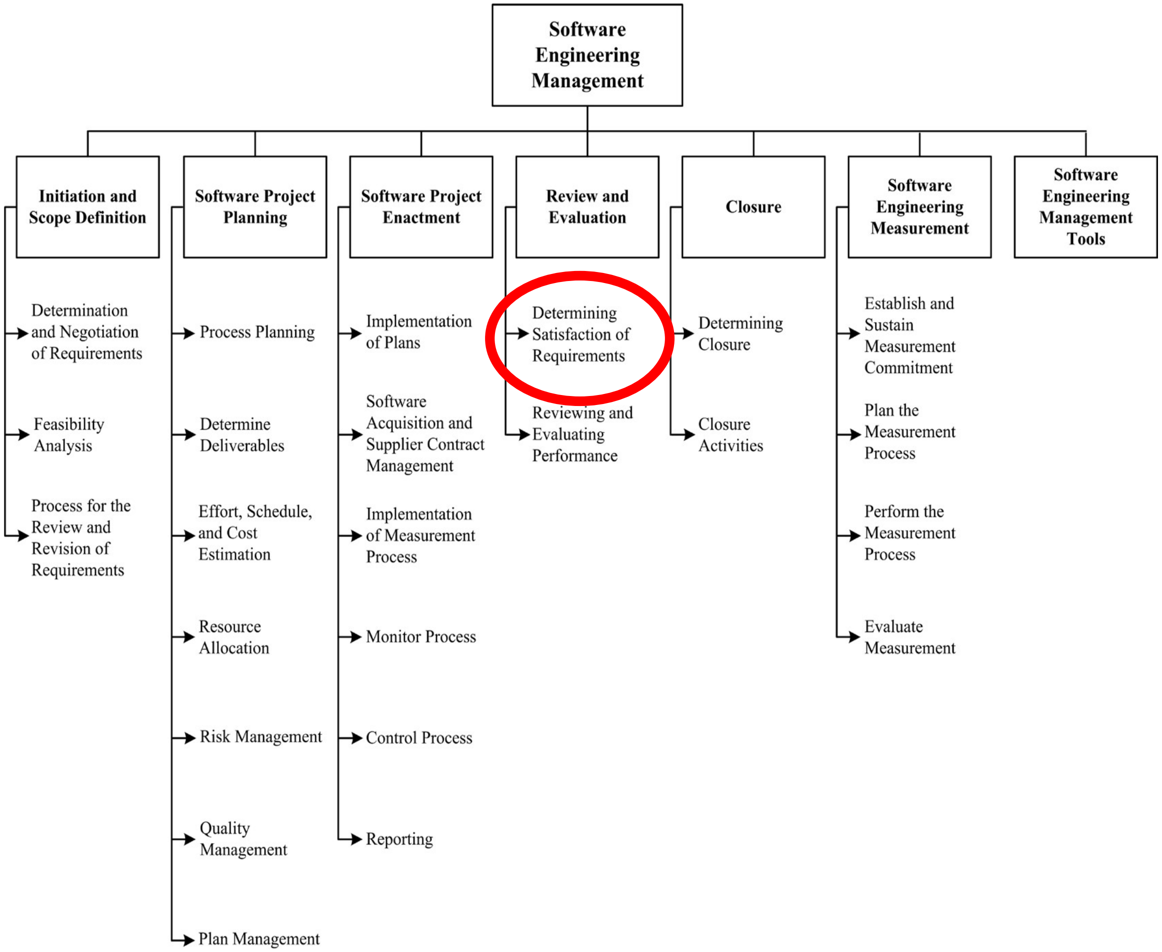
Software Configuration Management



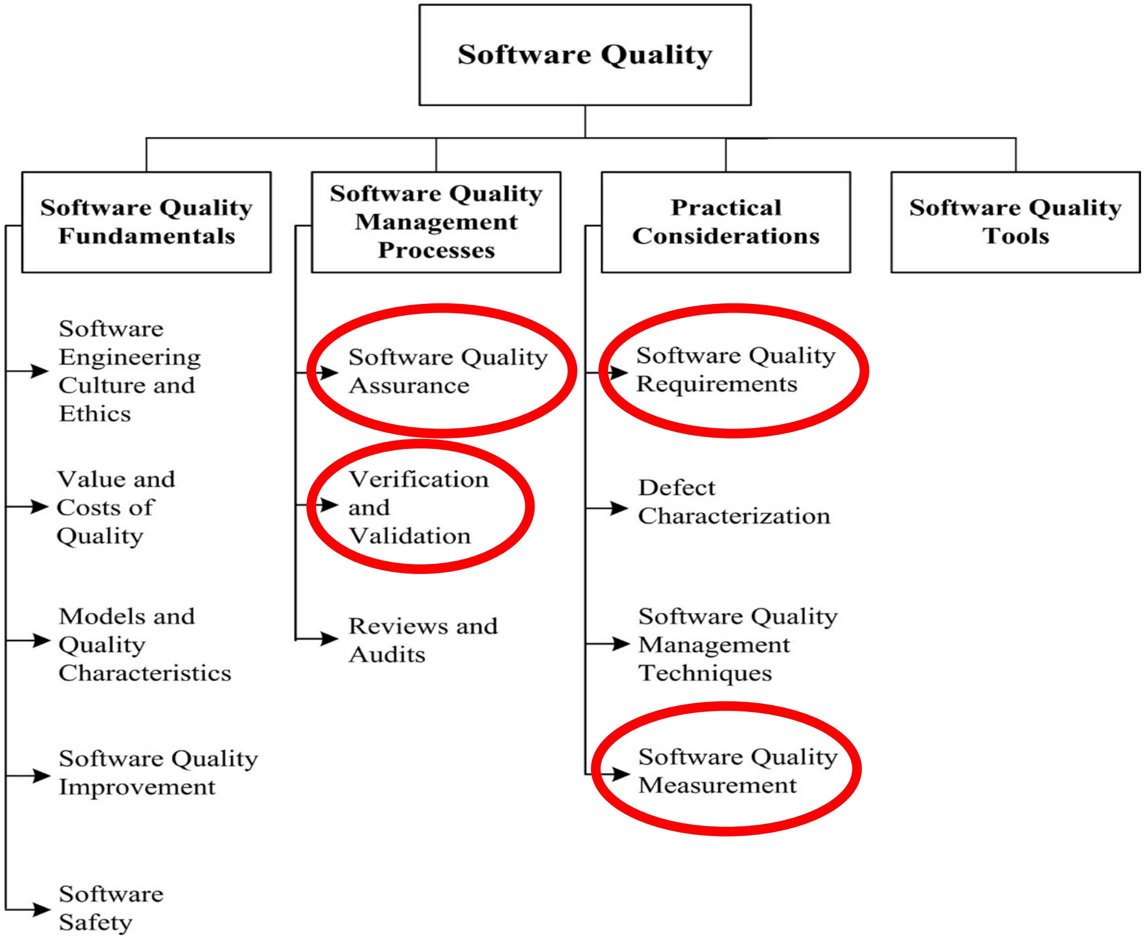
Software Configuration Management

- Management of the SCM Process
  - Organizational Context for SCM
  - Constraints and Guidance for SCM Process
  - Planning for SCM
  - SCM Plan
  - Surveillance of SCM
- Software Configuration Identification
  - Identifying Items to be Controlled
  - Software Library
- Software Configuration Control
  - Requesting, Evaluating, and Approving Software Changes
  - Implementing Software Changes
  - Deviations and Waivers
- Software Configuration Status Accounting
  - Software Configuration Status Information
  - Software Configuration Status Reporting
- Software Configuration Auditing
  - Software Functional Configuration Audit
  - Software Physical Configuration Audit
  - In-Process Audits of a Software Baseline
- Software Release Management and Delivery
  - Software Building
  - Software Release Management
- Software Configuration Management Tools

Software Engineering Management


Software Engineering Management
Initiation and Scope Definition
Determination and Negotiation of Requirements
Feasibility Analysis
Process for the Review and Revision of Requirements
Software Project Planning
Process Planning
Determine Deliverables
Effort, Schedule, and Cost Estimation
Resource Allocation
Risk Management
Quality Management
Plan Management
Software Project Enactment
Implementation of Plans
Software Acquisition and Supplier Contract Management
Implementation of Measurement Process
Monitor Process
Control Process
Reporting
Review and Evaluation
Determining Satisfaction of Requirements
Reviewing and Evaluating Performance
Closure
Determining Closure
Closure Activities
Software Engineering Measurement
Establish and Sustain Measurement Commitment
Plan the Measurement Process
Perform the Measurement Process
Evaluate Measurement
Software Engineering Management Tools

Software Quality


Software Quality
Software Quality Fundamentals
Software Quality Management Processes
Practical Considerations
Software Quality Tools
Software Engineering Culture and Ethics
Value and Costs of Quality
Models and Quality Characteristics
Software Quality Improvement
Software Safety
Software Quality Assurance
Verification and Validation
Reviews and Audits
Software Quality Requirements
Defect Characterization
Software Quality Management Techniques
Software Quality Measurement

What Do We Mean by Quality?

Concepts of Quality for Products

“Quality is *conformance to requirements*”
Crosby

“Quality is *fitness for intended use*”
Juran

“Quality is *value to someone*”
Weinberg

"Quality is Conformance to Requirements"

- If ***testable requirements*** can be established, then it is possible to decide ***whether the product satisfies the requirements*** – **by testing it**.
- If ***measurable quality characteristics*** can be established, then it is possible to decide on ***the extent to which the product satisfies the requirements*** – **by measuring it**.
- **Thus you can avoid disputes and have workable contractual relationships**

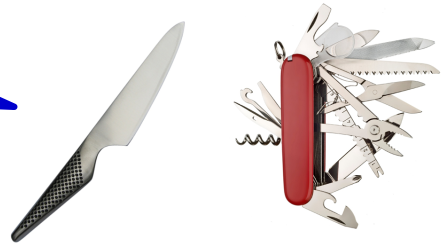
HOWEVER …

Issues with "Conformance to Requirements" (1 of 4)

Who establishes the requirements?

- ***Sponsor*** - The one who pays for the product
- ***End User*** - The one who will use the product
- ***Sales or Marketing*** - The one who will sell the product
- ***Engineering*** - The ones who will design and build it

What the end user wants



Planetgeek.ch

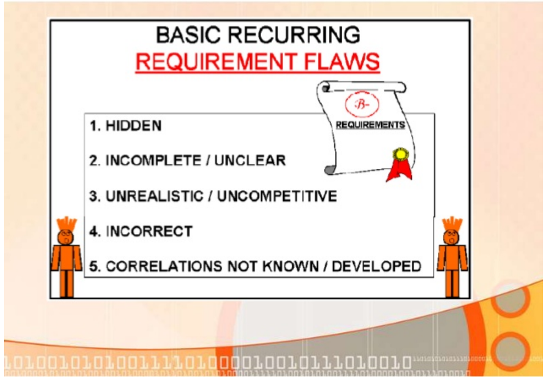
What the engineer builds

Issues with “Conformance to Requirements” (2 of 4)

Are the requirements right?

- consistent
- complete
- visible
- correct

Flaws



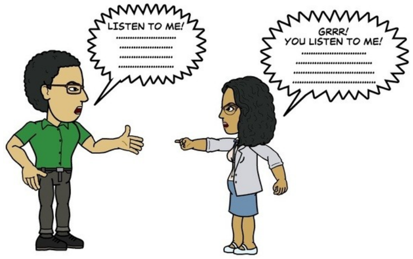
BASIC RECURRING
REQUIREMENT FLAWS

1. HIDDEN
2. INCOMPLETE / UNCLEAR
3. UNREALISTIC / UNCOMPETITIVE
4. INCORRECT
5. CORRELATIONS NOT KNOWN / DEVELOPED

REQUIREMENTS

Slideshare.net

- **Who determines whether the requirements are right?**



Quora.com

- **What if you discover a problem later on?**

Issues with "Conformance to Requirements" (3 of 4)

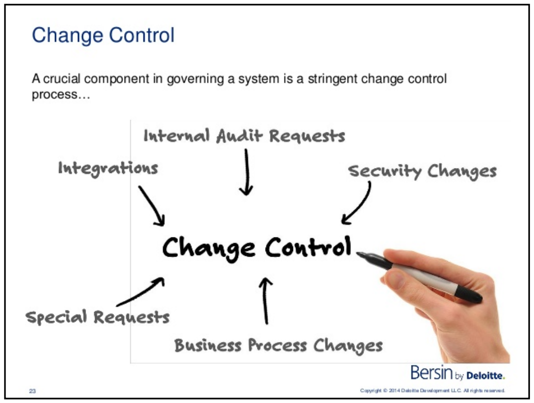
What about implicit vs. explicit requirements?

- ***Explicit requirement***: pizza should be hot and flavorful
- ***Implicit requirement***: **not harmful**

Issues with "Conformance to Requirements" (4 of 4)

What about when requirements change during the development process?

- ***Who makes the changes***?
- ***Who controls and authorizes the changes***?
- ***Who pays*** for the ***consequences*** of changes?



Change Control

A crucial component in governing a system is a stringent change control process…

23

“Quality is Fitness for Intended Use”

- **This definition is based on *a fundamental concept of law* - that *a product should be suitable* for the use that it is intended for.**
- **This definition accommodates the fact that *we may not be able to fully define the requirements*.**

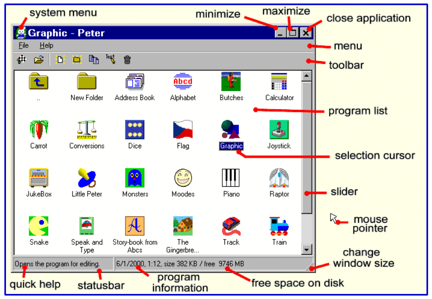
HOWEVER …

Issues with “Fitness for Intended Use” (1 of 3)

Who defines *fitness*?

- Consider a TV set
  - which fitness characteristics are not understood by
    - Typical User
    - Engineer
    - Sales Personnel
- Consider a software program
  - which fitness characteristics are not understood by
    - The typical software developer?
    - The typical end user?

Konga.com



Opens the program for editing.

6/1/2000, 1:12, size 382 KB / free 9746 MB

Gemtree.com

Issues with "Fitness for Intended Use" (2 of 3)

Different users have different definitions of fitness

- Ease of use for novices
- Control of fine details for experts
- Ease of maintenance for support staff
- Able to survive power failures
- Compatibility with previous system

Theodysseyonline.com

- **Uses change as users grow in experience**
  - Too many "ease of use" and "automatic" features may frustrate an expert

Issues with "Fitness for Intended Use" (3 of 3)

The "*pleasant surprise*" concept

User gets more than he or she expected

They really knew what they were doing when they designed this software

There is often tension between the engineer knowing better than the customer and the customer knowing better than the engineer

“Quality is Value to Someone”

- **This definition incorporates the idea that *quality is relative***
- **And it places increased emphasis on understanding *what quality means to the intended user* of the software**

HOWEVER …

Issues with "Value to Someone" (1 of 4)

Whose opinion counts?

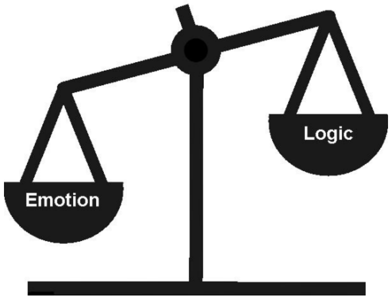
What features do you want?
Can it survive spilled drinks?
How is the financial software?
I want hot games
Does it have Facebook and Twitter?
dreamstime.com

- You may need to weigh different opinions

Issues with "Value to Someone" (2 of 4)

Logic vs Emotion

- "Glitz" v. "Substance"

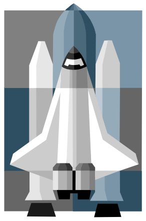


Which Car is Best for Our Family?

Issues with "Value to Someone" (3 of 4)

Value depends on What Features are Most Important

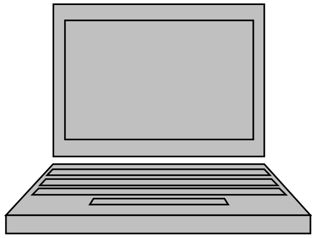
- Space Shuttle
  - 0 defects
  - Reliability



- Video Game
  - Good user interface
  - High performance

©Ron Leishman * illustrationsOf.com/439155

- School Laptop
  - Rugged
  - Fast
  - Good Battery Life
  - Good Software



Issues with "Value to Someone" (4 of 4)

Some Needs are Implicit (unstated)

Explicit

- I need an office
- It must have a computer
- And lots of space

Implicit

- I need a desk
- And a chair
- And convenient electrical outlets

Definitions of Software Quality

IEEE: The degree to which the software *possesses a desired combination* of *attributes*

Crosby: The degree to which a *customer perceives* that software *meets* composite *expectations*

Note that both definitions imply multiple expectations

Summary of Quality Definition Issues

- **You Must *Define Quality***
  - Before you can **engineer it** into your product
  - … and before you can **measure it**
  - … or **test** whether the product has the desired quality attributes

- **Quality has *Multiple Elements***
  - It reflects a multitude of expectations

- **Quality is *Relative***
  - Quality is in the eye of the customer

- **Quality encompasses *fitness, value, and other attributes***

Quality Attributes are Seldom Directly Measurable

- **Fitness for intended use**
- **Value to someone**
- **Satisfaction of requirements**
  - Including implicit, unstated requirements
- **Maintainability**
- **Reliability**
- **Supportability**
- **Testability**
- **…**

How can these be measured?

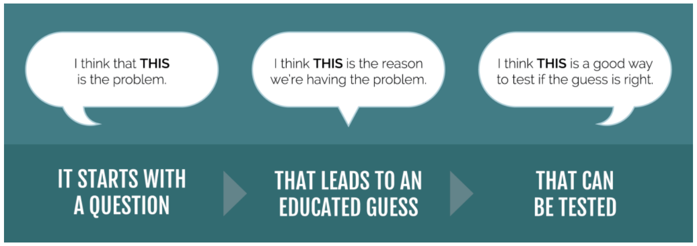
We need to find suitable *ways to measure* these attributes.

Some Attributes Are Measurable

Examples

- Water boils at 100^{o} Centigrade
- My new application will complete at least 10 searches per minute
- Code written in C takes less memory space than code written in Python

The above statements may or may not be true, but they can all be tested because they are all measurable.


I think that THIS is the problem.
I think THIS is the reason we're having the problem.
I think THIS is a good way to test if the guess is right.
IT STARTS WITH A QUESTION
THAT LEADS TO AN EDUCATED GUESS
THAT CAN BE TESTED

Elicitinsights.com

Some Attributes are Not Measurable

Examples:

- Joe’s code is better than Jan’s code
- Lisp is a superior programming language to C#
- Object oriented design produces code that is more maintainable

The above cannot be measured unless we define what we mean by:

- Better than
- Superior
- Maintainable

In a measurable way!

Surrogates

In order to measure an un-measurable attribute

- such as “quality” or “maintainability”

We may need to measure indirectly

- we measure something else that is associated with that attribute
- such as “defects” or “repair cost”

This alternative, measurable attribute is called a *surrogate*.

Surrogates Are Not the Real Thing

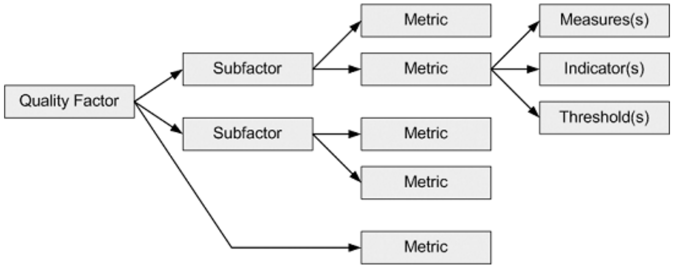
A surrogate may or may not accurately reflect the desired attribute

Examples:

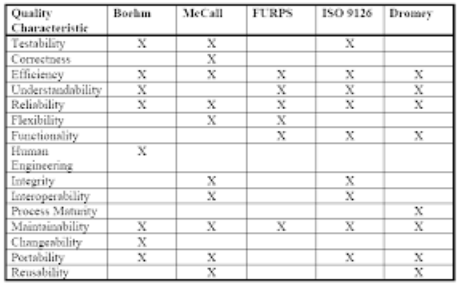
- **Defects** are a common surrogate for quality
- But lack of defects may or may not reflect **quality**.
  - Lack of defects may reflect failure to do effective testing
  - Or failure of the customer to use the product
- **Repair cost** may or may not reflect **maintainability** of the software
  - Perhaps "repair" included many changes to the software to add new features
  - Or perhaps the maintenance staff are not competent

There Are Systematic Ways to Identify Surrogates

- **Decomposition Approaches**
  - Fixed models
  - Individualized models



- **Standardized Approaches**
  - These enable comparisons of software from different organizations
  - But may not fit the desired quality characteristics of some software



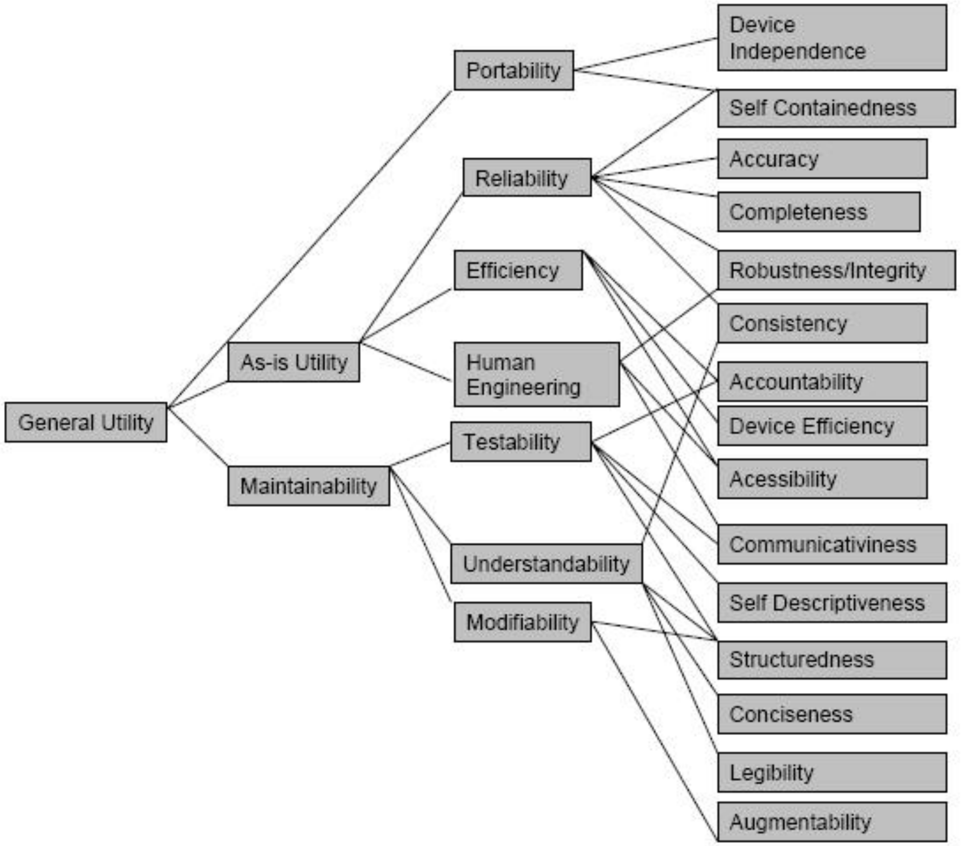
Quality Characteristic	Borhm	McCall	FURPS	ISO 9126	Dromey
Testability	X	X		X	
Correctness		X			
Efficiency	X	X	X	X	X
Understandability	X		X	X	X
Reliability	X	X	X	X	X
Flexibility		X	X		
Functionality			X	X	X
Human Engineering	X				
Integrity		X		X	
Interoperability		X		X	
Process Maturity					X
Maintainability	X	X	X	X	X
Changeability	X				
Portability	X	X		X	X
Reusability		X			X

Bvicam.ac.in

There is little consensus on how to measure quality attributes, so most organizations define them in ways that fit their specific customer needs.

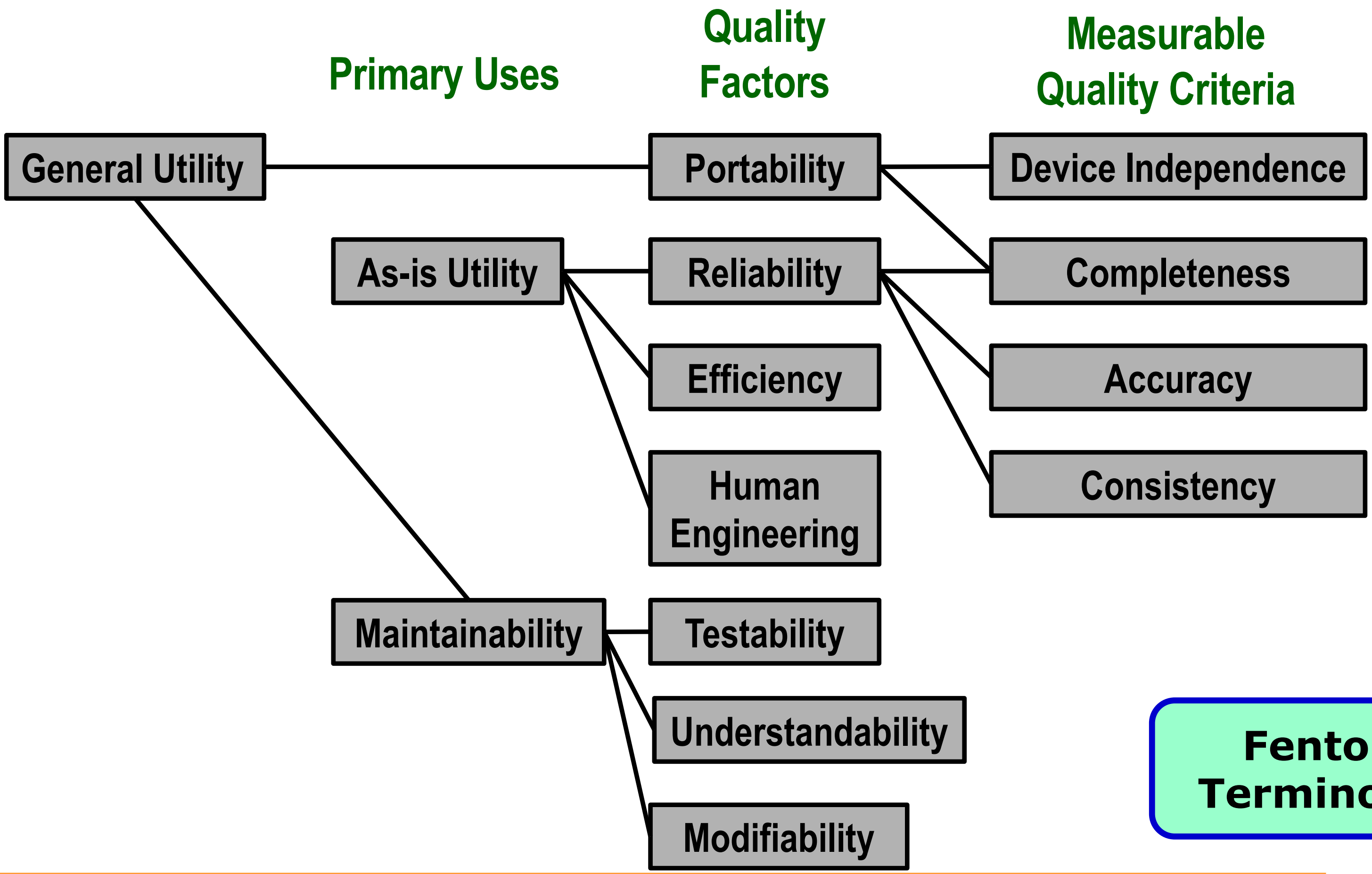
Decomposition Approaches

Boehm Software Quality Model



The concept here is to *decompose quality attributes or factors into subfactors* until you find factors that are measurable.

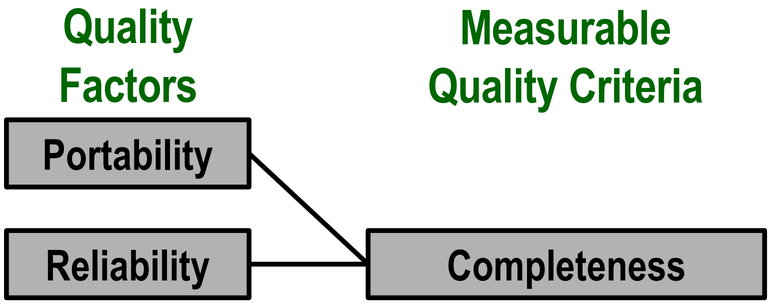
A Closer Look at the Boehm Model



Fenton’s Terminology

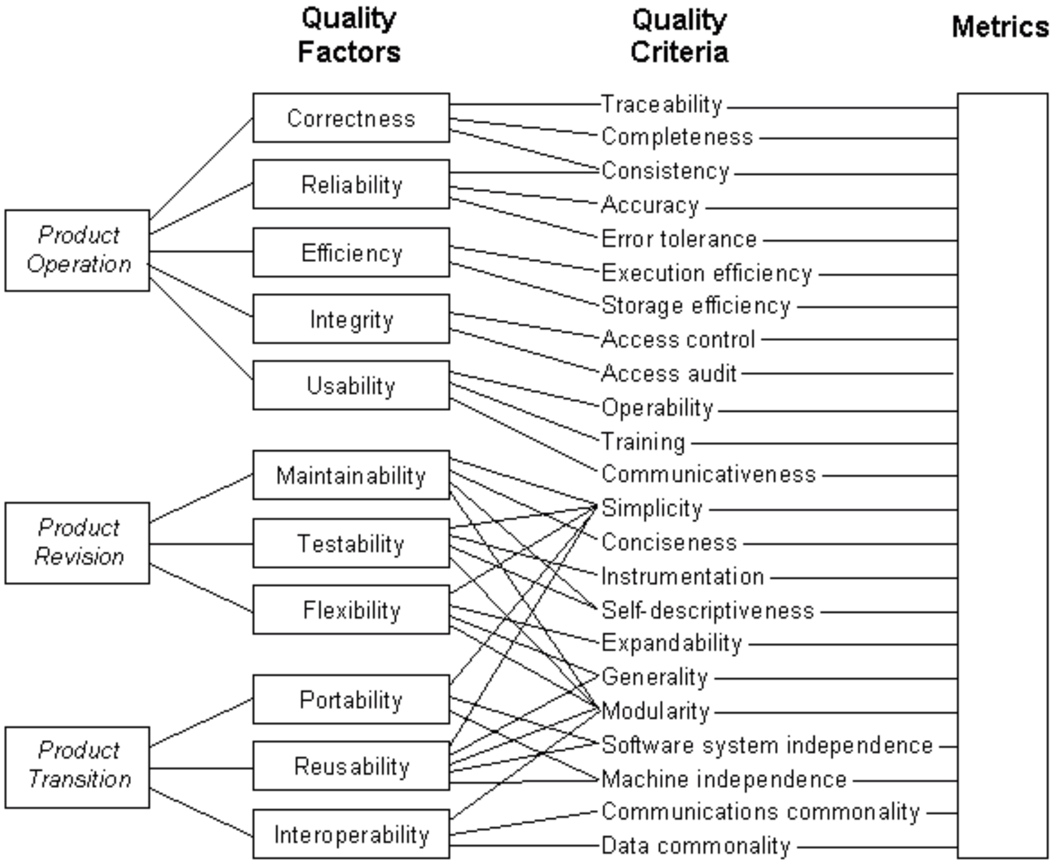
Comments on Boehm's Model

- **This is a way to *decompose what we mean by "quality"* until we have measurable attributes (quality criteria)**
- **These quality criteria are *surrogates* for quality**
  - There are many of them
  - Some of them relate to multiple quality factors



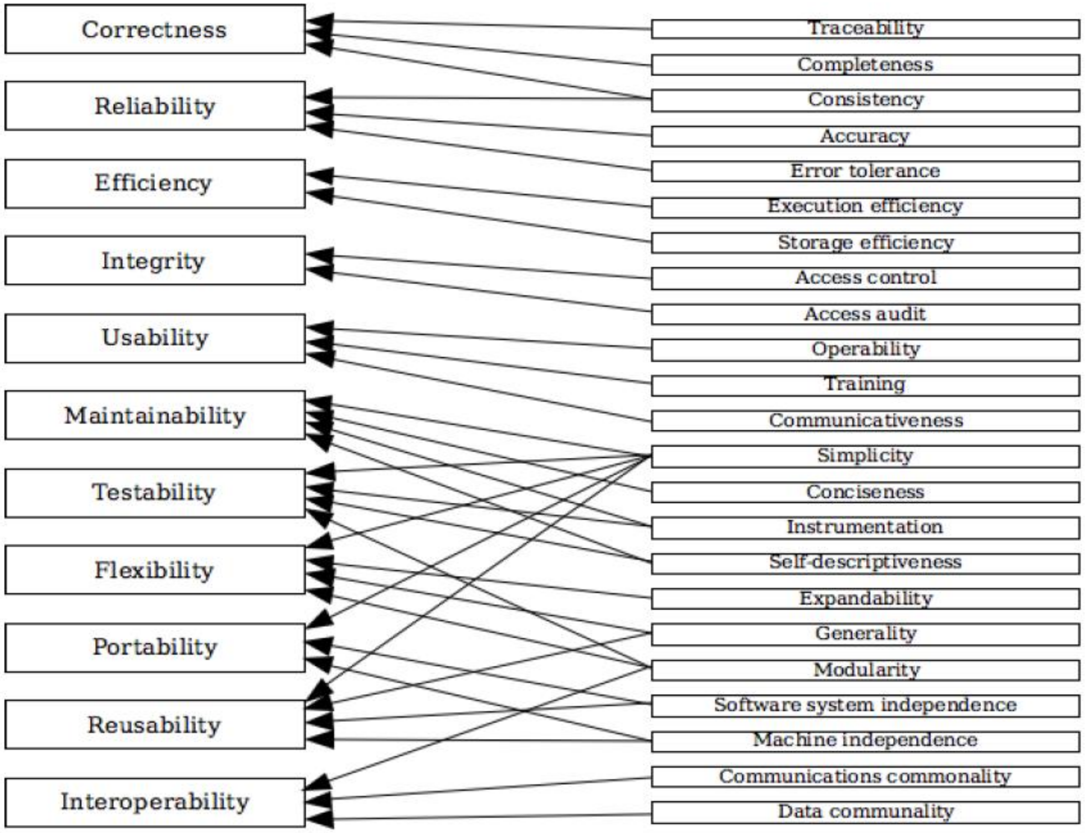
Decomposition Approaches

McCall Software Quality Model



As you can see, it's possible to establish a lot of criteria related to quality

McCall Model – Quality Factors and (Measurable) Criteria



As with the Boehm model, some criteria relate to multiple quality factors

Do I Really Need to Measure So Many Attributes?

- **The various models tend to be comprehensive**
  - But you may need to ***use only a portion*** of a model for your specific situation
  - Ultimately you need to ***measure only what will actually be used*** and be ***useful***

Measures of Software Quality

Based on

Defects or Faults or Failures

Quality = Lack of Defects (or Lack of Faults or Lack of Failures)[1]

The advantage of this approach is that it is often *easier to test for* defects or failures and *easier to measure* them than many other measures of quality

- However this approach ***may not capture what quality means to the end user***
  - Ease of use
  - Speed
  - …
- And it ***may not reflect all that the developer considers important***
  - Maintainability
  - Supportability
  - …

[1] **Defects and faults usually mean the same thing – causes of failures.**

Defect Density[1]

$$\textit{Defect Density} = \frac{\textit{Number of Defects}}{\textit{Size of Software Product}}$$

Variations:

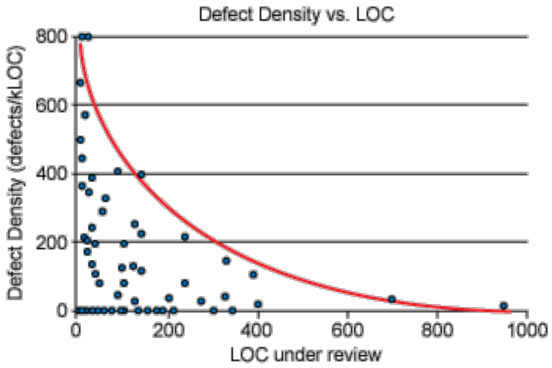
- *Failure Density* *(instead of defects)*
- *Number of Defects* *(this can be defined in different ways)*
  - *Known Defects*
  - $\textit{Total Defects}\ (\textit{Known Defects} + \textit{Latent Defects}^{2})$
- *Size of Software Product* *(can be defined in different ways)*
  - *It depends on the definition of size*

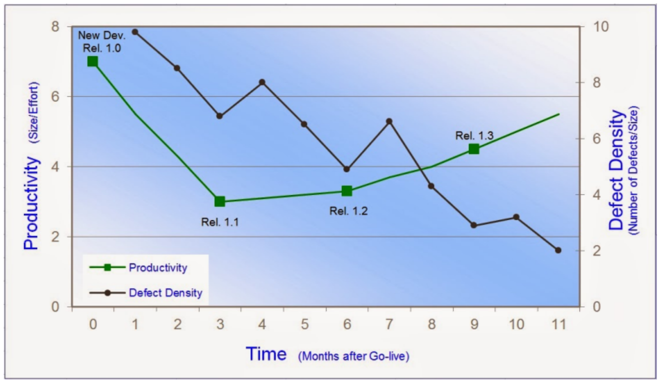
[1] Sometimes called "defect rate", although this is inaccurate

[2] Latent defects are defects we have not yet discovered

Defect Density Advantages

- **Easily measured, compared with other options**
- **Gives a good, *general idea of the overall quality* of the software**
- **This measure has been used for over 50 years to measure software, and overall the defect density has correlated well with perceived quality of products**





Defect Density Drawbacks
(1 of 3)

- **People can't always agree on *what constitutes a defect***
  - Failure in operation vs mistake in the code
  - Post-release defects vs defects found during development
  - Discovered vs latent defects
- ***Severity of problems* caused by defects may be *hard to assess***
  - Some software defects have no significant impact on customer's perception of quality
  - Different customers use the software in different ways

Example from IBM[1]

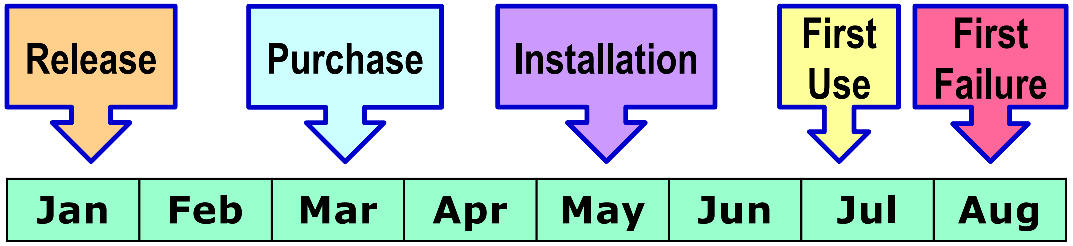
- **Approximately *one out of three defects* will only cause a user failure *once in 500 years*.**
- **A *very small portion* of defects (<2%) *cause the most important user failures***

Number of defects <u>may</u> <u>not</u> be strongly correlated to the frequency or severity of end user failures.

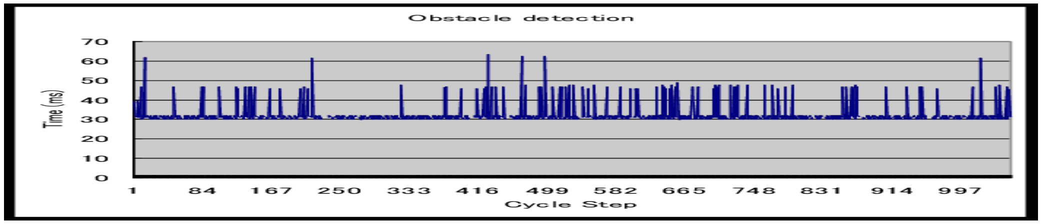
[1] See Adams in reference list.

Defect Density Drawbacks
(2 of 3)

- ***Different measures of the time scale***
  - Amount of ***time since release of product***
  - Amount of ***time the product is actually used***
  - ***Processing time*** actually used by the product



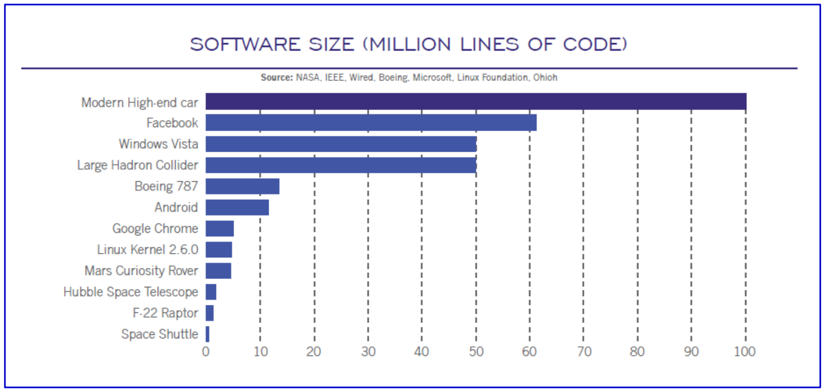
No problems with the software?
Not yet. But we only use it once a year.
shutterstock.com · 175562495



Researchgate.net

Defect Density Drawbacks
(3 of 3)

- ***Different measures of size***
  - This can make it hard to compare different projects or processes or development methods or organizations



SOFTWARE SIZE (MILLION LINES OF CODE)

Source: NASA, IEEE, Wired, Boeing, Microsoft, Linux Foundation, Ohioh

- ***What is defect density telling us*****?**
  - The quality of our product?

    or

  - The effectiveness of our defect detection and correction process?

Despite These Drawbacks, Defect Density is Very Widely Used

Some metrics that incorporate defect density

- Cumulative defect density
  - During development or after delivery
- Total serious defects found
- Mean time to fix serious defects
- Defects found during design reviews per KLOC
- Code inspection or peer review defects found per KLOC
- System test errors found per KLOC
- Customer-discovered problems per KLOC or per product

Usability
Hard to Test For & Hard to Measure

Formal Definition:

Usability is the degree to which a system can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.

ISO/IEC 25010 (2011)

Commonly used concepts of usability:

- User Friendliness
- Ease of use

This is a very complex concept that is hard to measure, but important to most end users

Three Categories of Usability[1]

- **Effectiveness**
  - Can users complete the tasks correctly?
  - Example: $\mathbf{Effectiveness} = \dfrac{\boldsymbol{Quantity * Quality}}{\mathbf{100}}$
- **Efficiency**
  - Time required to complete the tasks
  - Example: $\mathbf{Efficiency} = \dfrac{\boldsymbol{Effectiveness}}{\boldsymbol{Task\ Time}}$
- **Satisfaction**
  - Degree to which the end user likes the software
    - ***This is a very subjective measure***

[1] See Fenton, section 10.3 for further details

Internal Attributes Generally Viewed as Related to Usability

These are more readily measured and can be measured before the software is released

- Good use of menus
- Good use of graphics
- Good help functions
- Consistent interfaces
- Well-organized reference manuals and help files

Researchers have been unsuccessful in relating these to effectiveness, efficiency or customer satisfaction.

Use of these to predict usability is not recommended.

Testability

A product is testable if:

- It can be tested in a reasonable way (readily testable)
- The tests are well defined, comprehensive, and not overly redundant
- Each test can be directly traced to and from:
  - product requirements,
  - derived requirements resulting from design decisions, or
  - design or coding elements calling for specific testing
- Each test failure can be directly traced to:
  - a requirement that is not being met, or
  - A design element that was not properly implemented, or
  - A portion of the code that has a programming error

Good testing starts with testable requirements and designs.

Testing is unsuitable when …

- It would destroy the product
- It is too dangerous
- It is too costly
- It cannot reasonably be expected to provide confidence that requirements are satisfied
- It cannot be done

Evaluation Techniques
(other than testing)

- ***Examination***
  - For example, reading designs or code or other documents to check for errors
- ***Demonstration***
  - e.g. flying an airplane to show that it can fly
  - e.g. running a program to show that it works
- **Other techniques (examples)**
  - providing a ***formal proof*** that a program is correct
  - ***measuring*** something
  - showing through ***statistical analysis*** that the probability of a defect is below a threshold

Reasons why Requirements/Designs May be Hard to Test

- **Requirements may not be well understood**
- **Requirements may not be well documented**
- **What seems obvious to the customer or the system designer may not seem clear or obvious to the software developer or tester**
  - Different kinds of knowledge
  - Unstated assumptions
- **The customer and the software developer may not agree on what constitutes an acceptable test**
- **Changes made during software development may not be communicated to the software team**

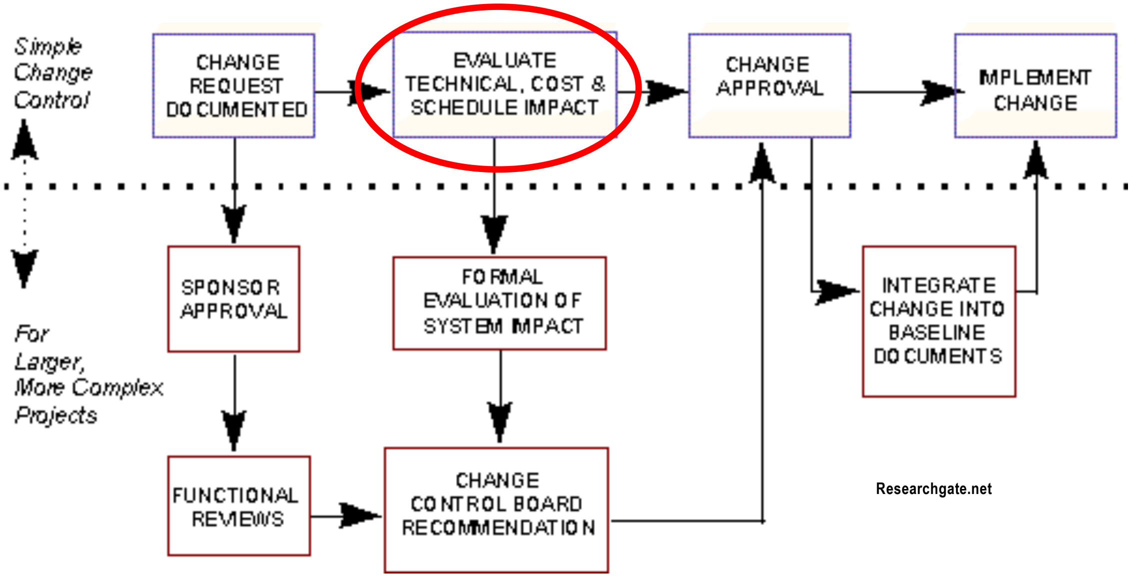
Suggestions (slide 1 of 3)

- **A requirement or design feature is not complete until you have <u>reached agreement on how it is to be tested</u>**
  - For each requirement, reach agreement between the software team and the customer or system engineer on how the requirement is to be tested
  - For each design feature, reach agreement between the software designer and the software test team on how the design feature is to be tested

www.cigniti.com

Suggestions (slide 2 of 3)

- **Control changes to requirements and design**
  – Don’t allow a requirements or design change without a clear understanding of the effect of the change on the software cost, schedule and technical development
  – For each change to requirements or design, indicate how the corresponding tests must be changed.

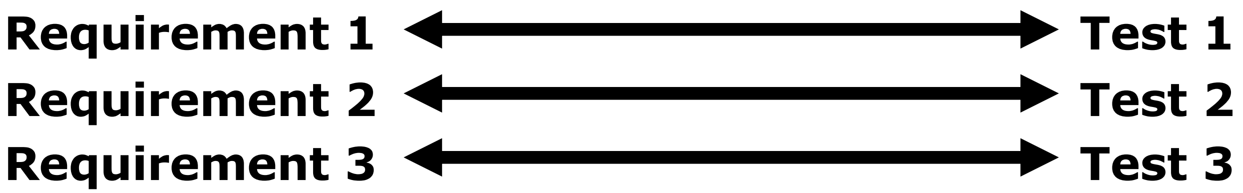

Simple Change Control
CHANGE REQUEST DOCUMENTED
EVALUATE TECHNICAL, COST & SCHEDULE IMPACT
CHANGE APPROVAL
IMPLEMENT CHANGE
SPONSOR APPROVAL
FORMAL EVALUATION OF SYSTEM IMPACT
INTEGRATE CHANGE INTO BASELINE DOCUMENTS
For Larger, More Complex Projects
FUNCTIONAL REVIEWS
CHANGE CONTROL BOARD RECOMMENDATION

Researchgate.net

Suggestions (slide 3 of 3)

- **Keep track of which tests correspond to which requirements or design elements *(traceability)***

Ideal

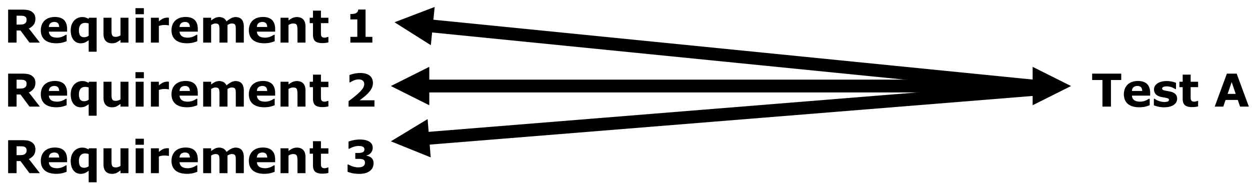


Requirement 1 ↔ Test 1

Requirement 2 ↔ Test 2

Requirement 3 ↔ Test 3

Acceptable



Requirement 1

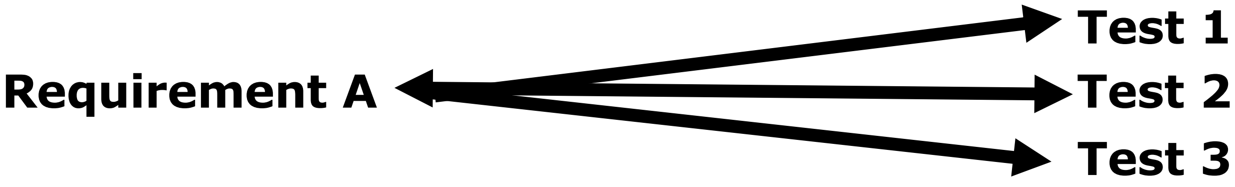
Requirement 2

Requirement 3

Test A

Other Traceability Options

Acceptable

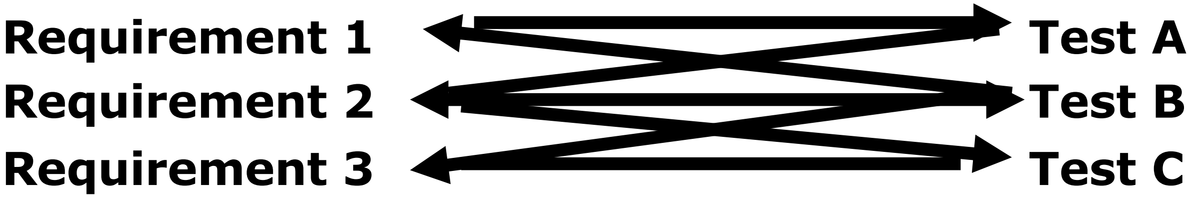

Requirement A

Test 1

Test 2

Test 3

Undesirable


Requirement 1

Requirement 2

Requirement 3

Test A

Test B

Test C

Reasons Why Code May Be Difficult to Test

- **Code is not well structured**
  - Needlessly complex
  - Poorly organized

We will address this in parts 4 and 5

- **Code elements do not trace directly to requirements or design elements**
  - So when the code causes a failure, it is hard to determine whether the problem is with the code or the design or the requirement

- **Code is not well documented or does not follow coding conventions**
  - Hard to understand
  - Error prone

Seeding and Tagging
A simple and effective way to assess Testing Progress

Seeding and Tagging

Purpose: **To help you estimate how many undetected errors (defects) are in your code**

When to do this: **During test planning and during the testing process**

***Suppose*: You have been testing your code and have discovered D_1 errors (defects).**

***Question*: How many errors are left?**

***Technique*: Seeding and Tagging**

***Concept*: Introduce extra errors and see how many of them your test process has found.**

Overview

1. **Inject extra errors**

   before testing starts

2. **See how many of those errors you find during the normal testing process**

Seeding and Tagging Details

- **Introduce a given number of extra errors into the software -- say *E* of them**
- **Run standard tests, detecting D_2 of them**
- **Compute D_2/E = % of errors detected**
- **Suppose D_1 = number of genuine errors already detected**
- **Then you assume the total number of errors in the software is**

$$\boxed{D_1 * E / D_2}$$

Example of Seeding and Tagging

- **200 defects found so far**
- **You have injected 20 extra defects**
- **You have found 12 of these extra defects**
- **Therefore, assume total defects =**

200 * 20 / 12 = 4000 / 12 = 333 total defects

=> 333 - 200 = 133 defects remaining

By performing this analysis from time to time, you can estimate your defect density and your testing progress over time.

Part 2

Some Basic Principles of Measurement and Data Analysis

Contents

- **Definitions**
- **Scales**
- **Basic Analysis Approaches**
- **Statistical Distributions**
- **Other Statistical Concepts**

Definition of Measurement

Measurement is …

… the process by which numbers or symbols are assigned to <u>attributes</u> of entities in the real world in such a way so as to describe them according to clearly defined rules.

- **The assignment of numbers must <u>preserve</u> intuitive and empirical <u>observations</u> about the attributes and entities**

Source: Fenton, page 5

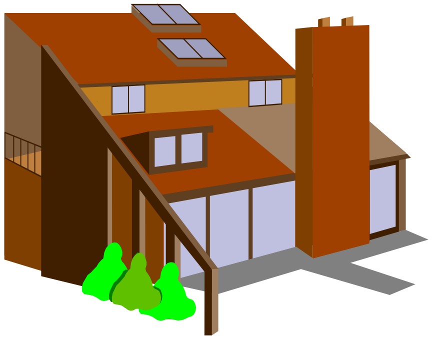
Preservation of Attributes Example

"House A is bigger than House B"

This is a meaningful statement only if the number we assign to "size" preserves our intuitive notion of houses and their sizes.

House A

House B



But Intuitions Vary

What do we really mean by "size"?

Before we can measure size, we must define a *model* that reflects a specific *viewpoint*

- The model must specify an **entity** to be measured

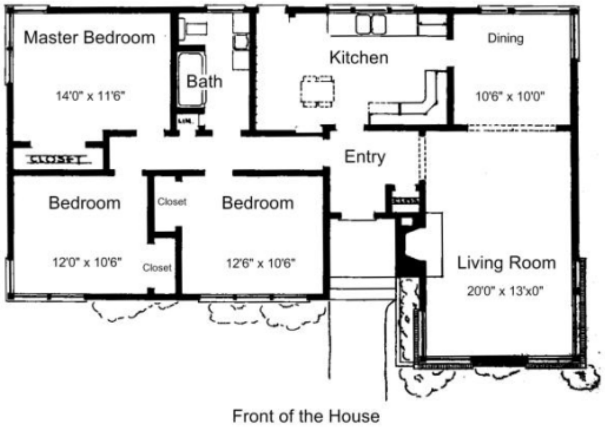
and

- an **attribute** of that entity.
- I.e.,
  - **what do you want to measure?** and
  - **what do you want to know about it?**
- Examples (next two slides)

Functionality Model for Size of a House

In this model, the size of a house is based on ***how many people can comfortably live there***

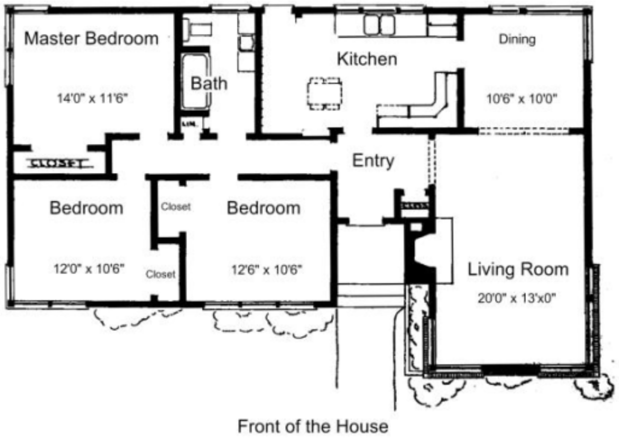
- The size of a house is measured by the ***number of bedrooms and the number of bathrooms***



Cost Model for Size of a House

In this model, the size of a house is based on ***how much it will cost to construct it***

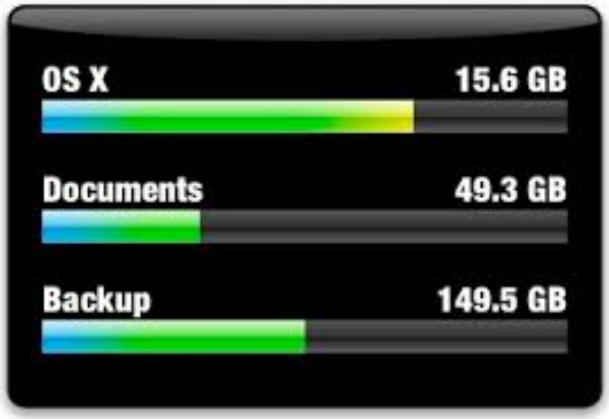
- The size of a house is measured by the ***square feet of living space***



Storage Space Model for Size of Software Used on a PC

In this model, the size of software is based on *how much disk space it occupies*

- The size of software is measured by the ***number of bytes of memory*** it requires when stored on a disk

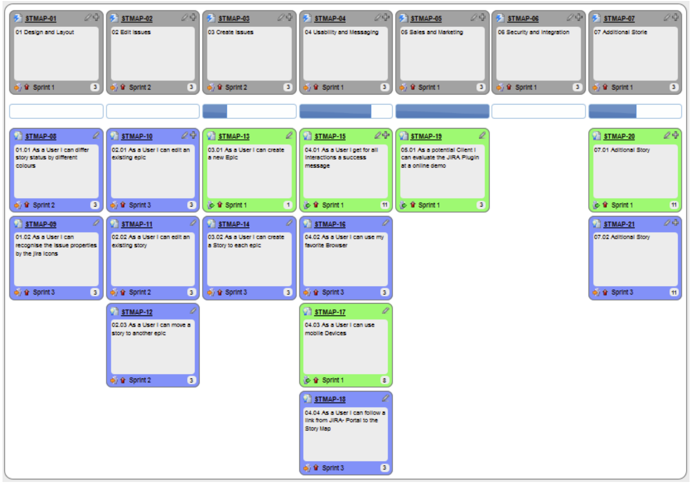


https://encrypted-tbn2.gstatic.com/images?q=tbn:ANd9GcSGaF3LbJOReKic-6yZyOr9_cKS3Lr3gV0zzEX9rsMezUxHinFliw

Cost Model for
Size of Software Used on a PC

In this model, the size of software is based on *how much it will cost to construct it*

- The size of software is measured by the ***number of user stories*** used to establish the requirements.



STMAP-01
01 Design and Layout
Sprint 1 3

STMAP-08
01.01 As a User I can differ story status by different colours
Sprint 2 3

STMAP-09
01.02 As a User I can recognise the issue properties by the jira icons
Sprint 3 3

STMAP-02
02 Edit Issues
Sprint 2 3

STMAP-10
02.01 As a User I can edit an existing epic
Sprint 3 3

STMAP-11
02.02 As a User I can edit an existing story
Sprint 2 3

STMAP-12
02.03 As a User I can move a story to another epic
Sprint 2 3

STMAP-03
03 Create Issues
Sprint 2 3

STMAP-13
03.01 As a User I can create a new Epic
Sprint 1 1

STMAP-14
03.02 As a User I can create a Story to each epic
Sprint 3 3

STMAP-04
04 Usability and Messaging
Sprint 1 3

STMAP-15
04.01 As a User I get for all interactions a success message
Sprint 1 11

STMAP-16
04.02 As a User I can use my favorite Browser
Sprint 3 3

STMAP-17
04.03 As a User I can use mobile Devices
Sprint 1 8

STMAP-18
04.04 As a User I can follow a link from JIRA- Portal to the Story Map
Sprint 3 3

STMAP-05
05 Sales and Marketing
Sprint 1 3

STMAP-19
05.01 As a potential Client I can evaluate the JIRA Plugin at a online demo
Sprint 1 3

STMAP-06
06 Security and Integration
Sprint 1 3

STMAP-07
07 Additional Storie
Sprint 1 3

STMAP-20
07.01 Aditional Story
Sprint 1 11

STMAP-21
07.02 Aditional Story
Sprint 3 11

https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwikptj9kdfSAhWKz4MKHfSNAfEQjRwIBw&url=https%3A%2F%2Fen.wikipedia.org%2Fwiki%2FUser_story&bvm=bv.149397726,d.cGc&psig=AFQjCNFdWIGXMDrGnfsbmrgiNdbLFEGADg&ust=1489620239765305

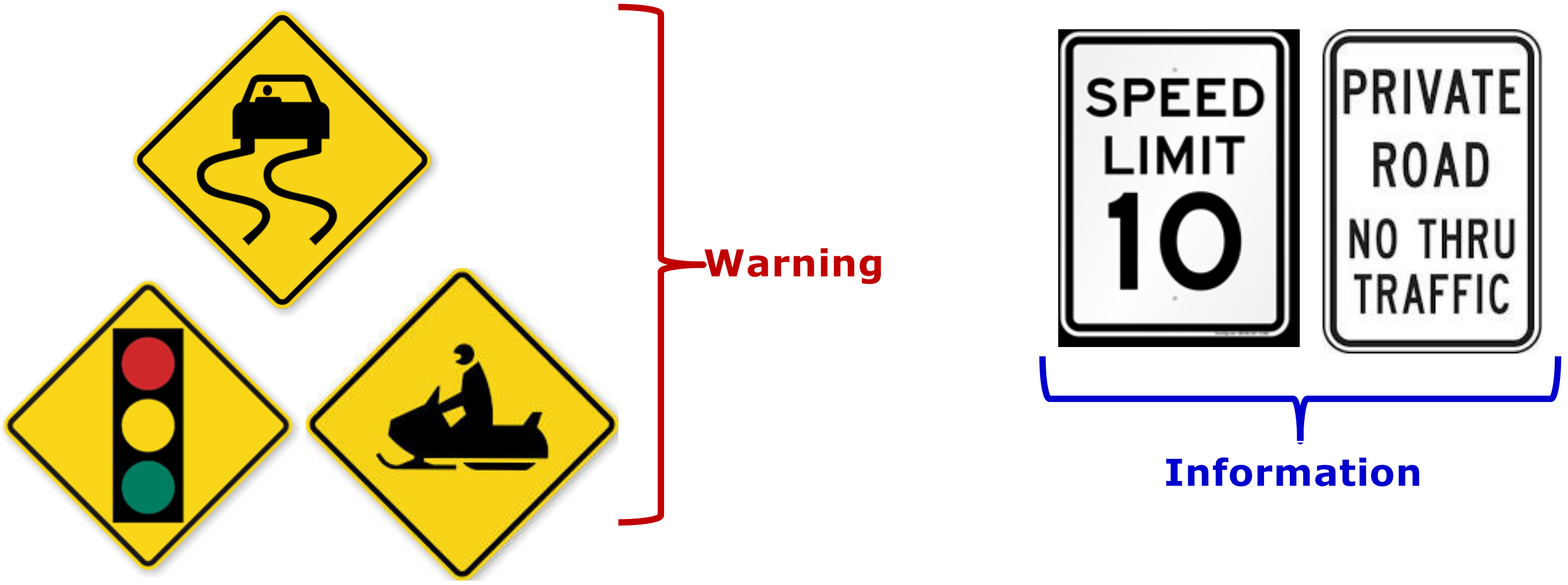
Contents

- **Definitions**
- **Scales**
- **Basic Analysis Approaches**
- **Statistical Distributions**
- **Other Statistical Concepts**

Measures Allow Us to Categorize and Analyze the Attributes of Entities

But measures do not have to be numbers.

Example: Road Signs – shape and color are used to categorize road signs



Warning

Information

Definitions: Scales of Measure

A *scale of measure*

or

A *level of measurement*

or simply

A *scale,* is:

- A ***collection of symbols or numbers*** used to **classify** attributes of entities or variables
- A ***classification system*** for **describing** the nature of information

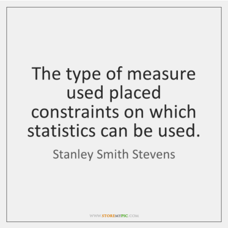
➢ **There are various scales in use and the properties of those scales are important**

- The properties help determine which forms of analysis are appropriate

Classification of Scales

The classification system most widely adopted in data analysis and statistics was originated by Stanley Smith Stevens[1] for use in psychological research.

www.fixquote.com



The type of measure used placed constraints on which statistics can be used.

Stanley Smith Stevens

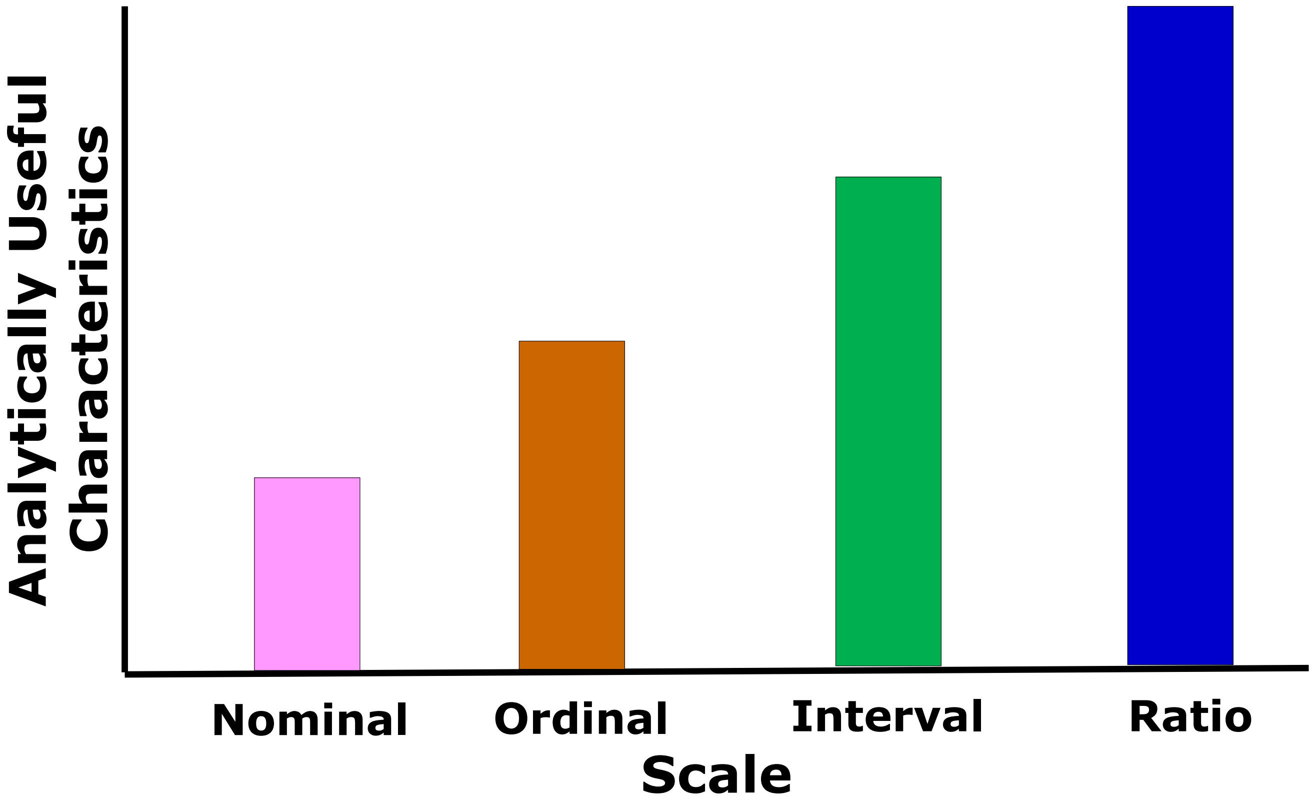
www.storemypic.com

- **This classification includes four levels of scales:**
  - Nominal
  - Ordinal
  - Interval
  - Ratio

Researchers have debated Stevens' classification and proposed other classification systems, but no other scale classification has achieved such widespread use.

[1] *Stevens, S. S. (7 June 1946). "On the Theory of Scales of Measurement". Science. 103 (2684): 677–680.*

Stevens' Types of Scales



Scales Help Us Understand What Kinds of Analysis are Meaningful

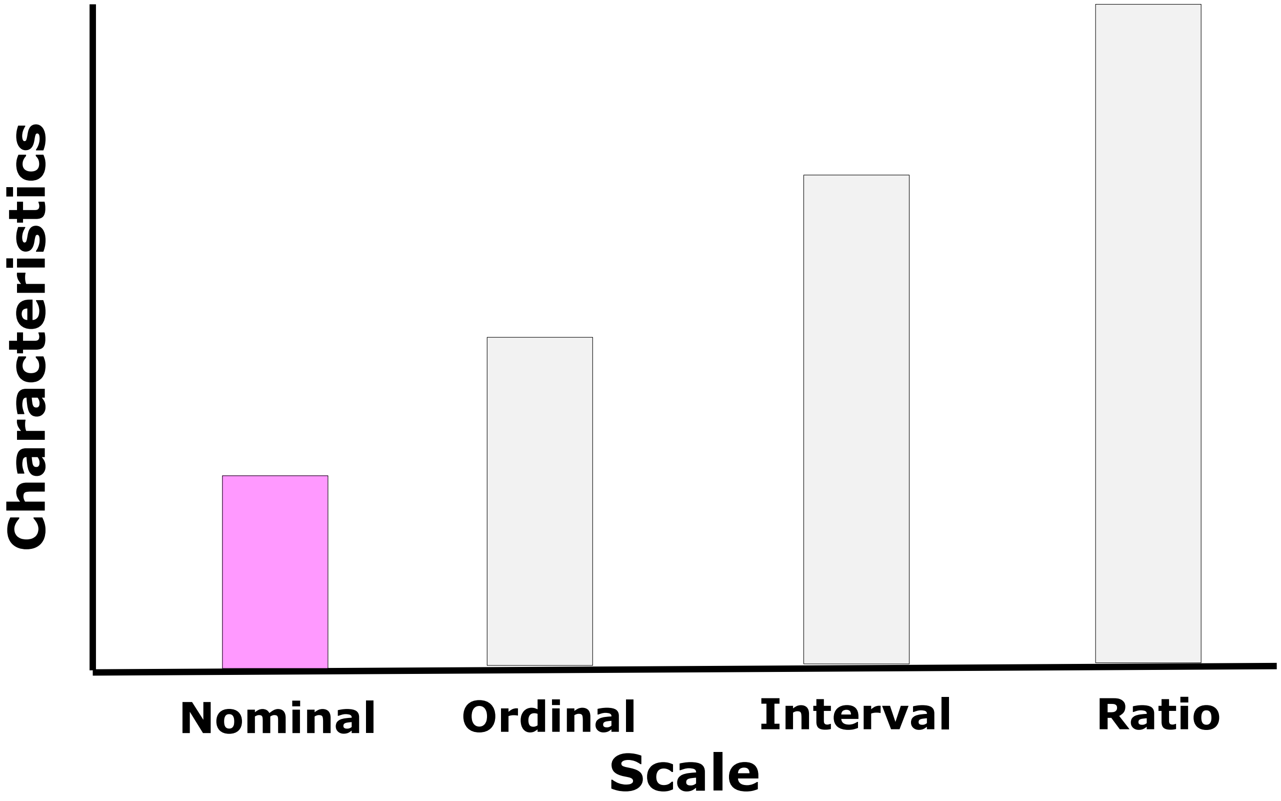
Color

No Natural Order

Height

Natural Order

Nominal Scales



Nominal: a scale that places entities in ***categories*** – but without any ordering

Example: Color

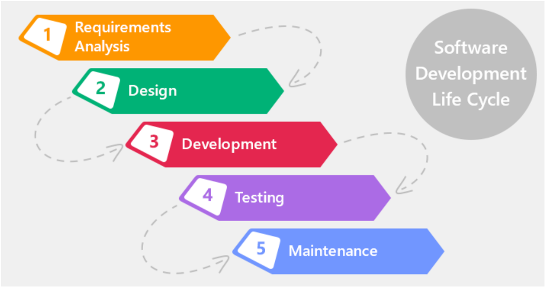
– marbles are

- Blue, or
- Black, or
- Red, or
- Yellow

Software Example: Defect Origin

- defects discovered while testing are

- Requirements related, or
- Design related, or
- Programming related, or
- Testing related.



www.xbsoftware.com

Nominal Scales - Characteristics

- ***We can categorize the entities***
  - *Often in terms of specific qualities such as color or shape*
- ***We can count the size or frequency of each category***
- ***We can determine the most frequent category (the mode)***
- ***But there is no natural order or ranking to the categories***
- ***And there's no such thing as the median or average***

Other Examples of Nominal Scales

- Gender
- Nationality
- Ethnicity
- Language
- Genre
- Style
- Biological Species
- Form or Shape
- Parts of Speech (in grammar)

We can use numbers to classify attributes or variables in a nominal scale

But they do not have any numerical values or relationships other than equality or inequality

Example:
Suppose We Have Three Animal Species

1. **Tigers**

Mynicetime23.wordpress.com

Just because we have numbered them in a particular order does not mean anything.

2. **Elephants**

True-wildlife.blogspot.com

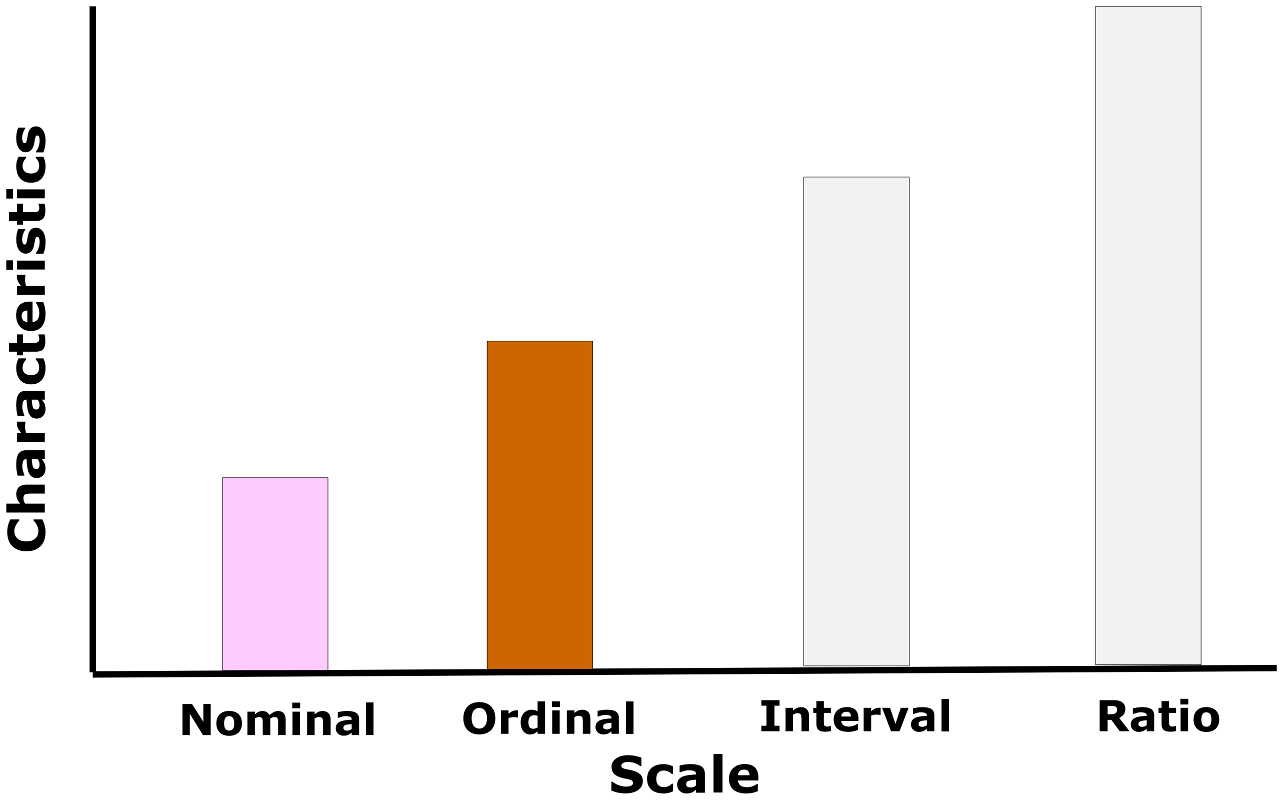
Tigers are not more important because they are listed first

3. **Horses**

kidssearch.com

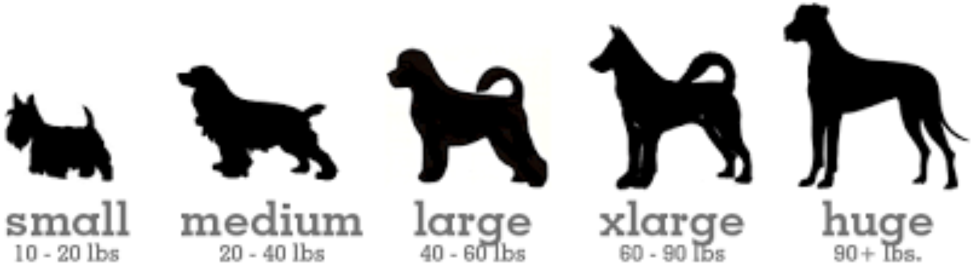
Horses are not 3 times as valuable because we gave them the number three

Ordinal Scales



Ordinal: there is a ***ranking*** or ***ordering***

- **Example: military rank**
  - General
  - Colonel
  - Lieutenant
  - Sergeant
- **SW Example: defect severity**
  - minor
  - significant
  - major
- **Example: size of dogs**

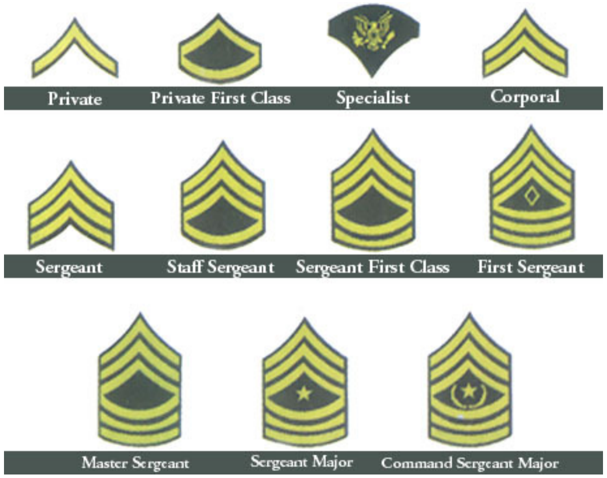


https://cdn.shopify.com/s/files/1/0118/8082/files/dog_sizes_antler.png?373

Ordinal Scales - Characteristics

- You can place them in *order* (sorting)
- You can determine the *median* (middle) item in a list

➢ But you cannot compute an average



➢ And there is no mathematical relationship between categories

➢ (sergeant is not "twice as" high as "private")

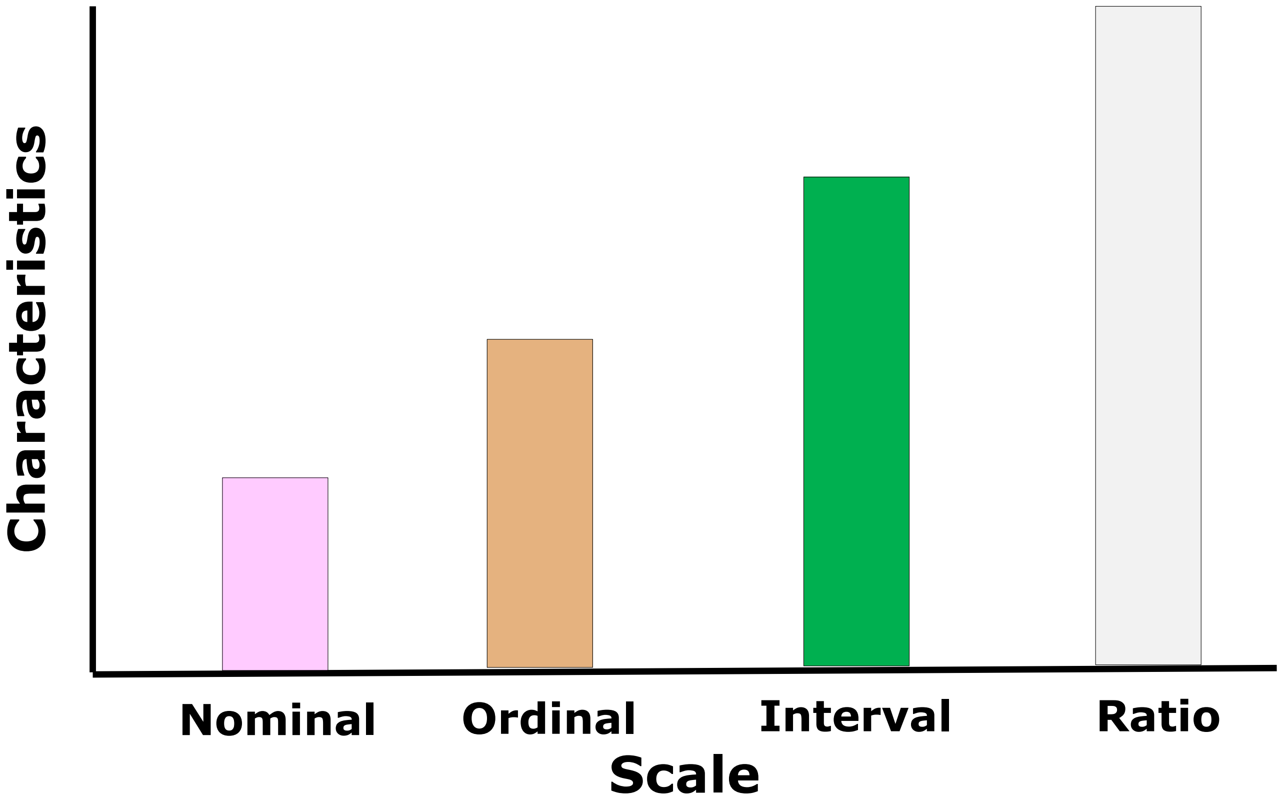
Ordinal Scales – Other Examples

- **Rank in a contest or race**
- **Degree of health**
  - Critical Serious Fair Good Excellent
- **Comparative restaurant ratings**
  - Excellent Very Good Good Poor Terrible

The degree of distance between successive categories is not defined

You can determine the middle item (median) but not the "average" (mean)

Interval Scales



Interval: there is a ***fixed distance*** between consecutive members of a sequence

- **Example: Dates**
  - 1/1/2012, 1/2/2012, …

Given any two dates, we can count the distance between them in days

- **Example: Temperature in Farenheit**
  - 10 degrees, 30 degrees, etc.

Given any two temperatures, we can count the distance between them in degrees

Interval Scales - Characteristics

- There is an *ordering*

You can tell if one is larger than another

- You can quantify the *distance* (degree of difference) between any two values

Given any two dates, we can count the distance between them in days

- You can *add or subtract* values

30º plus 10º = 40º

3/5/2020 – 3/1/2020 = 4 days

➢ But you **cannot multiply or compute a ratio**

  – See next slide

With Interval Scales We Cannot Multiply or Divide

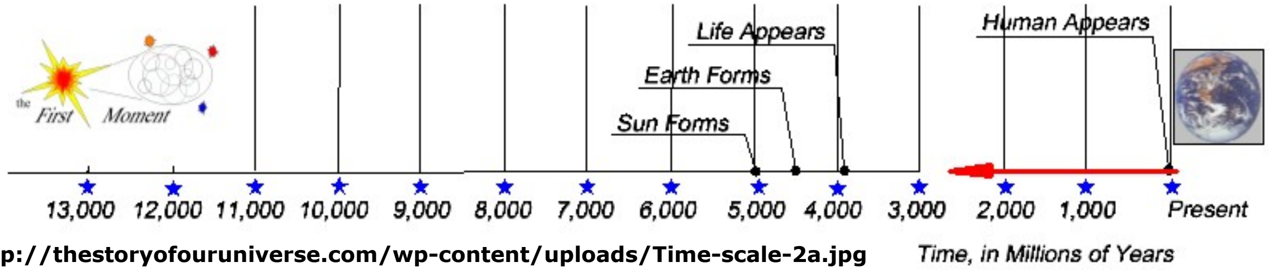
- Example: Dates
  - 1/1/2012, 1/5/2014, 1/25/2020, …

It does not make sense to say one date is "five times" another date

Mathematically, this means there is no "true 0" value -- we can place the "0" value anywhere in the sequence

Interval Scale Example - Time

- **We lack a precise measure of when time began**
- **So we measure time using an arbitrary "0" point**
  - Years are measured since an arbitrary date
    - Microsoft Excel dates are measured in days since 12/31/1899
  - Scientists often measure time in years backwards from the present



http://thestoryofouruniverse.com/wp-content/uploads/Time-scale-2a.jpg

Other Examples of Interval Scales

- Temperature (on Centigrade or Fahrenheit scale)
- Map coordinates (longitude or latitude)
- Map direction (degrees from North)

Note: a common error is to assume you can compute ratios for items in an interval scale.

See next slide for more information on this.

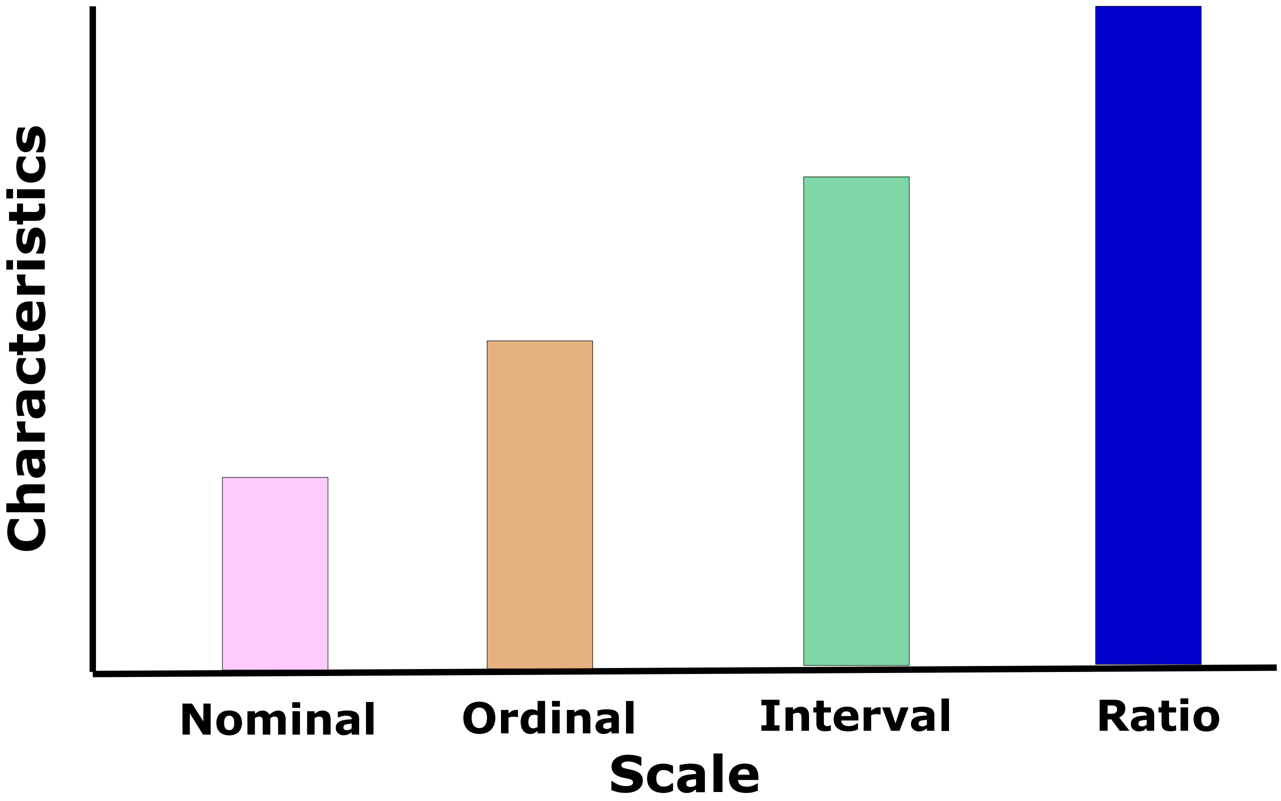
Computing Ratios for Interval Scales

- **The ratio between two items on an interval scale cannot be determined**
- **But you can determine the ratios between differences**

Example: the parent's age could be twice that of the child's:

Each age is the difference between the current date and the date of birth

Ratio Scales



Ratio: There is a fixed distance between consecutive sequence members, AND *multiplication is meaningful*

This means that ratios are meaningful

- **Examples:**
  - Size
  - Weight
  - Length
  - Duration
  - Electric Charge

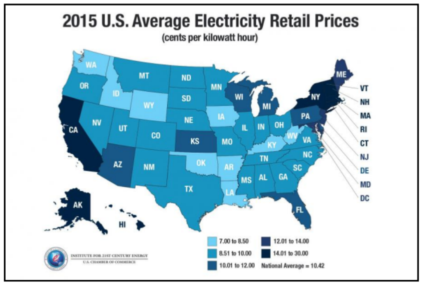
http://www.sixsigmatrainingconsulting.com/wp-content/uploads/2010/10/ratio-scale.bmp

60 mph is twice as fast as 30 mph

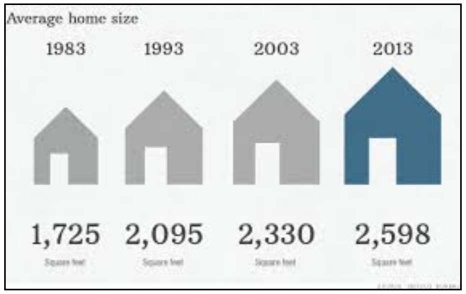
Note that there is a "true 0" value

With a Ratio Scale We Can Compute an Average or Mean

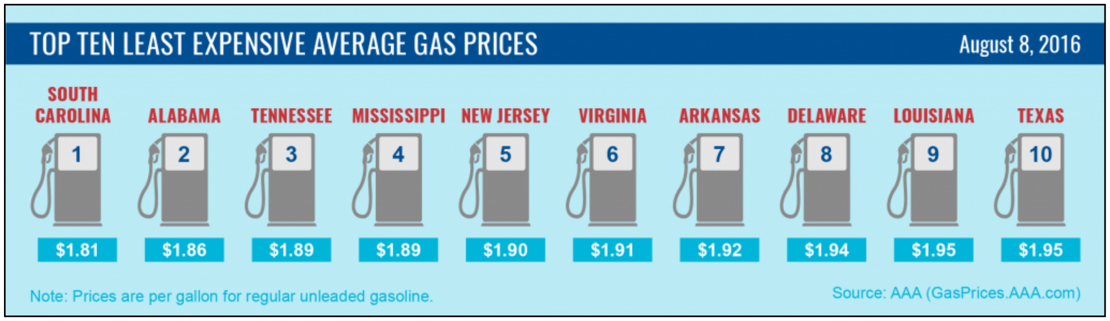
- **Because we can multiply and divide**



https://www.uschamber.com/sites/default/files/styles/article_gallery/public/ei_electricityratesmap4.5.16.jpg?itok=DDSLVlpu



http://homevestors.com/wp-content/uploads/Average-Home-Size.png



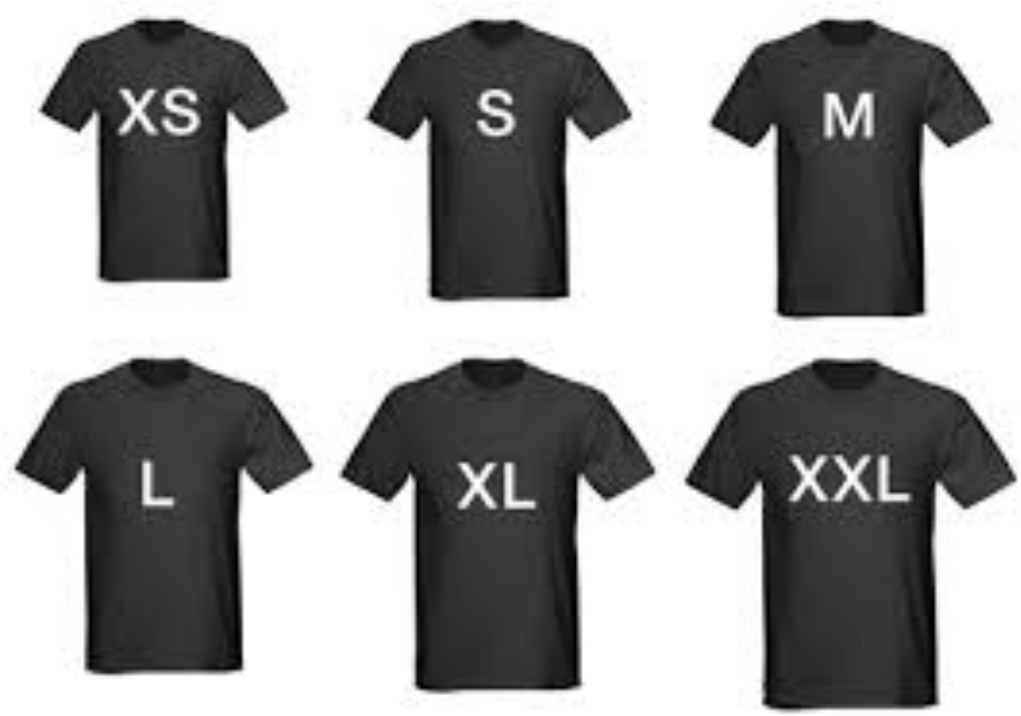
https://www.google.com/url?sa=i&rct=j&q=&esrc=s&source=images&cd=&cad=rja&uact=8&ved=0ahUKEwiXhcuLzNTSAhVM34MKHQ9UCNYQjRwIBw&url=http%3A%2F%2Fnewsroom.aaa.com%2F2016%2F08%2Faverage-gas-prices-holding-steady-begin-august%2F&psig=AFQjCNFYj2pTtSvClpmXIAwWf_1pFq-aeg&ust=1489532668700886

One Other Scale is Often Used in Data Analysis

Absolute Scales

- ***Absolute*: all mathematical operations are meaningful**
  - Square root
  - Exponentiation
  - Etc.
- **In some definitions, only positive values are permitted**
  - (i.e., there is an absolute 0, marking the starting point of the scale)

Test Yourself – What Scale is Clothing Size?



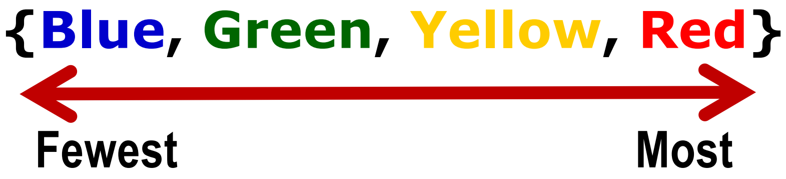
https://rules.ssw.com.au/PublishingImages/size-stories-bad-example.jpg

Scales - Summary

Characteristics	Nominal	Ordinal	Interval	Ratio
Categorization, Classification	☑	☑	☑	☑
Mode (most frequent)	☑	☑	☑	☑
Order, Comparison, Sorting		☑	☑	☑
Median (middle)		☑	☑	☑
Fixed Distance, Add, Subtract			☑	☑
Ratio of differences			☑	☑
Multiply and Divide, Ratio				☑
True 0				☑
Mean				☑

Misuse of Scales

Example - Assigning a Scale to Test Failures



{Blue, Green, Yellow, Red}

Fewest

Most

This is an ordinal scale

- It provides a ranking but not ratios
- There is not a fixed difference between values
  - The difference between "red" and "yellow" is not comparable to the difference between "yellow" and "green"
- It makes no sense to add, subtract, multiply or divide the values.

But Suppose we Replace with a Numeric Scale

4 = Blue

3 = Green

2 = Yellow

1 = Red

We are tempted to make meaningless or misleading statements like these:

"The average test error improved from 2.2 to 3.1"

"The average test error improved by 47%"

Another Example – Customer Survey

The average response from our customers is "good" [on a scale of very poor, poor, good, very good]

- But the scale is not a ratio scale (or even an interval scale), so what does "average" mean?
- Does "half very good and half poor" mean "good"?

Customer Survey
What do you think about the quality of our offerings?
EXCELLENT GOOD AVERAGE POOR
How would you rate our service?
EXCELLENT GOOD AVE
What are the ch
recommend us to others?
EXCELLEN AVERAGE POOR

http://www.brecoflex.com/wp-content/uploads/2016/10/survey.jpg

Assigning Numbers Can be Misleading

The properties of the number system may not necessarily apply to the attribute being measured

Consider the attribute "temperature":

Scale	Yesterday	Today
Centigrade	0	18
Fahrenheit	32	64

Is it twice as hot today as it was yesterday?

Twice as …

“Twice as” is a meaningful concept for numbers

It is not necessarily a meaningful concept for temperature

- Because centigrade and Fahrenheit are interval scales, not ratio scales

The error we make is assuming that properties of the number system apply to the attribute being measured

Footnote: “twice as” is a meaningful concept for temperature measured on the Kelvin scale because the volume of a gas is proportional to its temperature on that scale.

A Very Common Example: Student Grade Point Average

Student 1

- **B – 3 points**
- **B – 3 points**
- **B – 3 points**
- **B – 3 points**
- **B – 3 points**

- **Average: 15/5 = 3.0**

Student 2

- **A – 4 Points**
- **A – 4 Points**
- **B – 3 Points**
- **B – 3 Points**
- **D – 1 Point**

- **Average: 15/5 = 3.0**

Grade Point Average is an example of a ***Descriptive Statistic***.

Not technically accurate, but does give a useful indication.

Are These Students Really Equal?

Contents

- **Definitions**
- **Scales**
- **Basic Analysis Approaches**
- **Statistical Distributions**
- **Other Statistical Concepts**

What Do We Have, After Data Collection?

We have a collection of data values, known as a *dataset* or a *batch*.

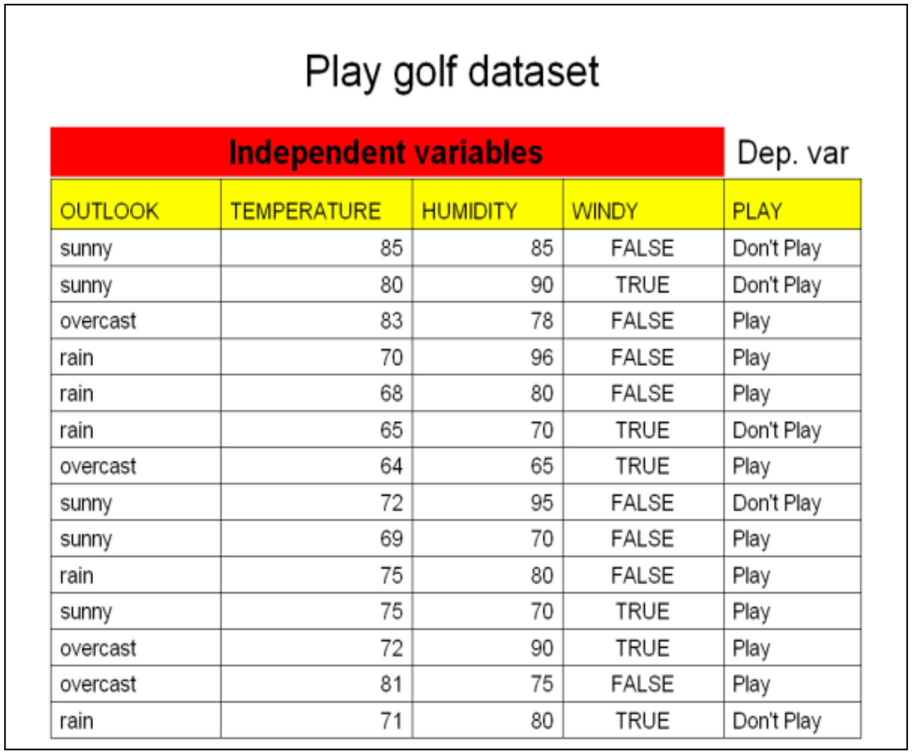
These values are *measurements*

of

attributes

of

entities



Play golf dataset

Independent variables				Dep. var
OUTLOOK	TEMPERATURE	HUMIDITY	WINDY	PLAY
sunny	85	85	FALSE	Don't Play
sunny	80	90	TRUE	Don't Play
overcast	83	78	FALSE	Play
rain	70	96	FALSE	Play
rain	68	80	FALSE	Play
rain	65	70	TRUE	Don't Play
overcast	64	65	TRUE	Play
sunny	72	95	FALSE	Don't Play
sunny	69	70	FALSE	Play
rain	75	80	FALSE	Play
sunny	75	70	TRUE	Play
overcast	72	90	TRUE	Play
overcast	81	75	FALSE	Play
rain	71	80	TRUE	Don't Play

What Do We Want to Know?

Characteristics **of attributes / entities of the same type**

Relationships **between attributes / entities of the same or different types**

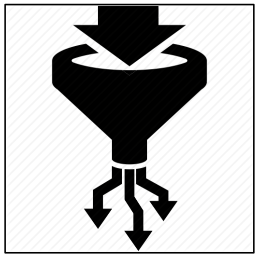
Examples:

- Which method is faster?
- How many defects?
- Do effective peer reviews reduce customer complaints?

The First Step is to Organize the Data for Analysis

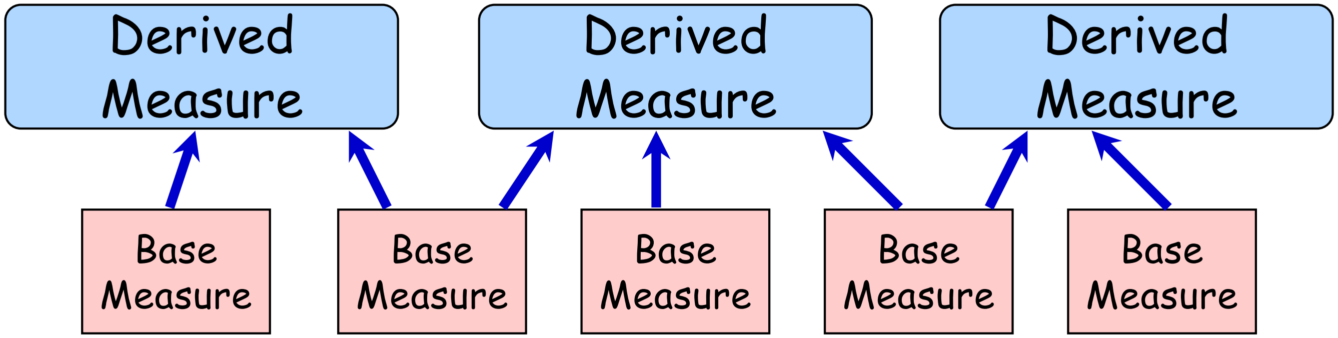
First, we <u>refine</u> & <u>compress</u> the data

- Eliminate duplicates, errors, etc.
- Compute totals, sort data, etc.
- Perform appropriate data compression



Thenounproject.com

Then, we compute various <u>derived measures</u> (computations)



Derived Measure

Derived Measure

Derived Measure

Base Measure

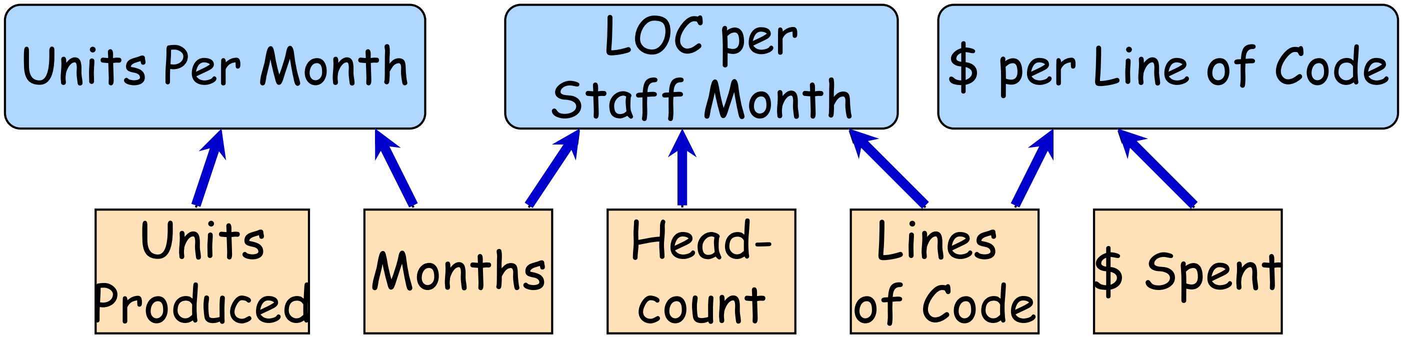
Base Measure

Base Measure

Base Measure

Base Measure

Examples of Derived Measures

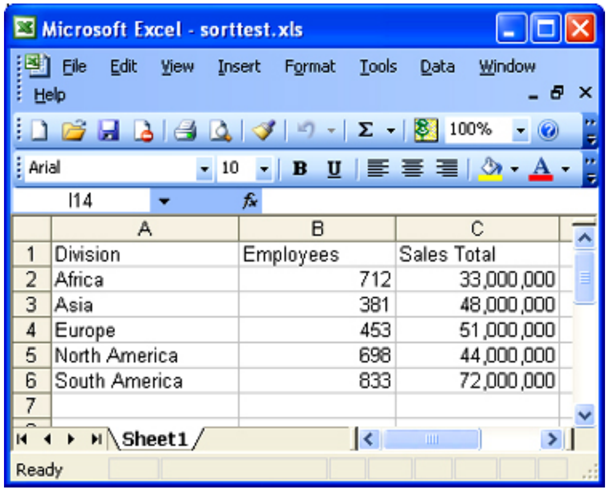


Exactly what derived measures we want is determined by our metric selection process (based on our information needs).

The Next Step is Analysis

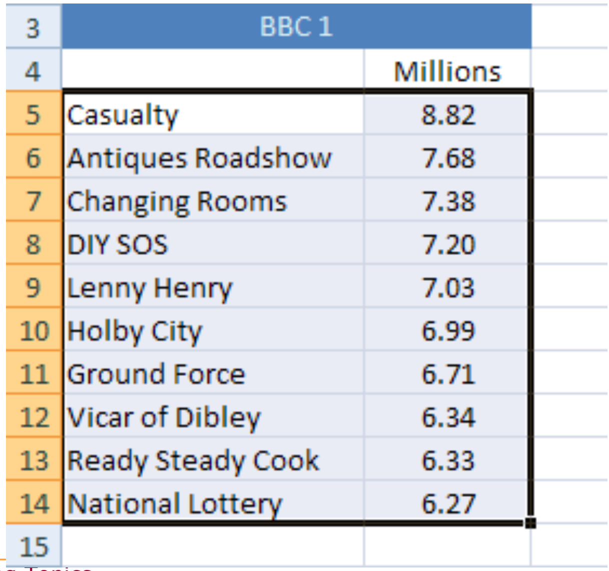
One option is to simply look at the data:

We can look at the data as we have it



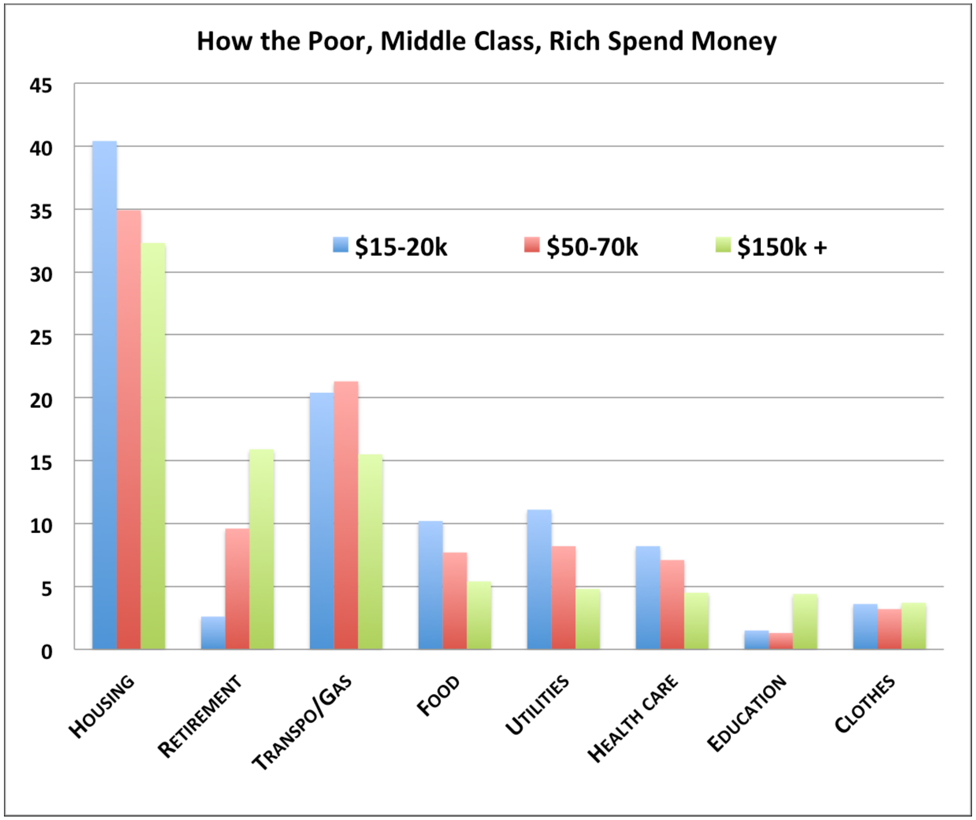
	A	B	C
1	Division	Employees	Sales Total
2	Africa	712	33,000,000
3	Asia	381	48,000,000
4	Europe	453	51,000,000
5	North America	698	44,000,000
6	South America	833	72,000,000
7			

We may want to sort the data or do other simple processing to make sense of it



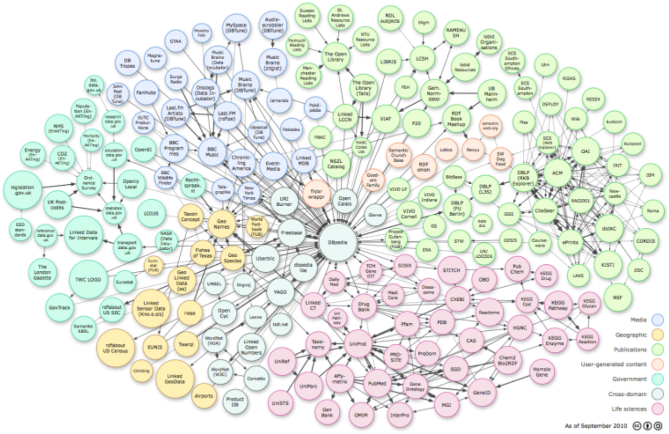
3	BBC 1	
4		Millions
5	Casualty	8.82
6	Antiques Roadshow	7.68
7	Changing Rooms	7.38
8	DIY SOS	7.20
9	Lenny Henry	7.03
10	Holby City	6.99
11	Ground Force	6.71
12	Vicar of Dibley	6.34
13	Ready Steady Cook	6.33
14	National Lottery	6.27
15		

Another Option is to Graph the Data

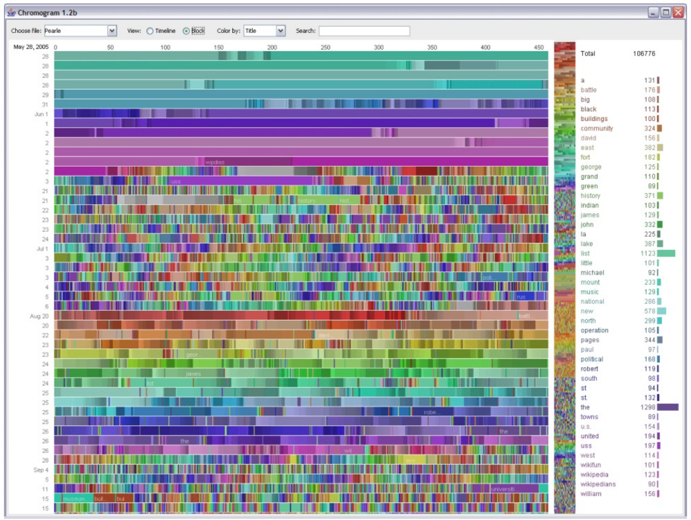


Often, a graph or chart makes it very easy to see what the data are telling us.

But Often There is Too Much Data or the Relationships Are Not So Easy to See



Statistical methods are often helpful in situations like these



Choose file: Pearle

View: ○ Timeline ● Block

Color by: Title

Search:

Total	106776
a	131
battle	176
big	108
black	113
buildings	100
community	324
david	156
east	382
fort	182
george	125
grand	110
green	89
history	371
indian	103
james	129
john	332
la	225
lake	387
list	1123
little	101
michael	92
mount	233
music	129
national	286
new	578
north	299
operation	105
pages	344
paul	97
political	168
robert	119
south	98
st	94
st.	132
the	1298
towns	89
u.s.	154
united	194
uss	197
west	114
wikifun	101
wikipedia	123
wikipedians	90
william	156

Contents

- **Definitions**
- **Scales**
- **Basic Analysis Approaches**
- **Statistical Distributions**
- **Other Statistical Concepts**

Statistical Techniques Help Us Deal with Many Situations

Statistical techniques **can often be used to describe the attributes and relationships**

Examples:

- High and Low values (and how often each occurs)
- Average (Mean), Median and Mode
- Mathematical relationships between values
  - For example, if you double the number of programmers, how much do you reduce the schedule?
- Patterns of values
  - For example, do most defects result from errors in requirements, errors in design, or errors in coding?
- Overall distribution of values

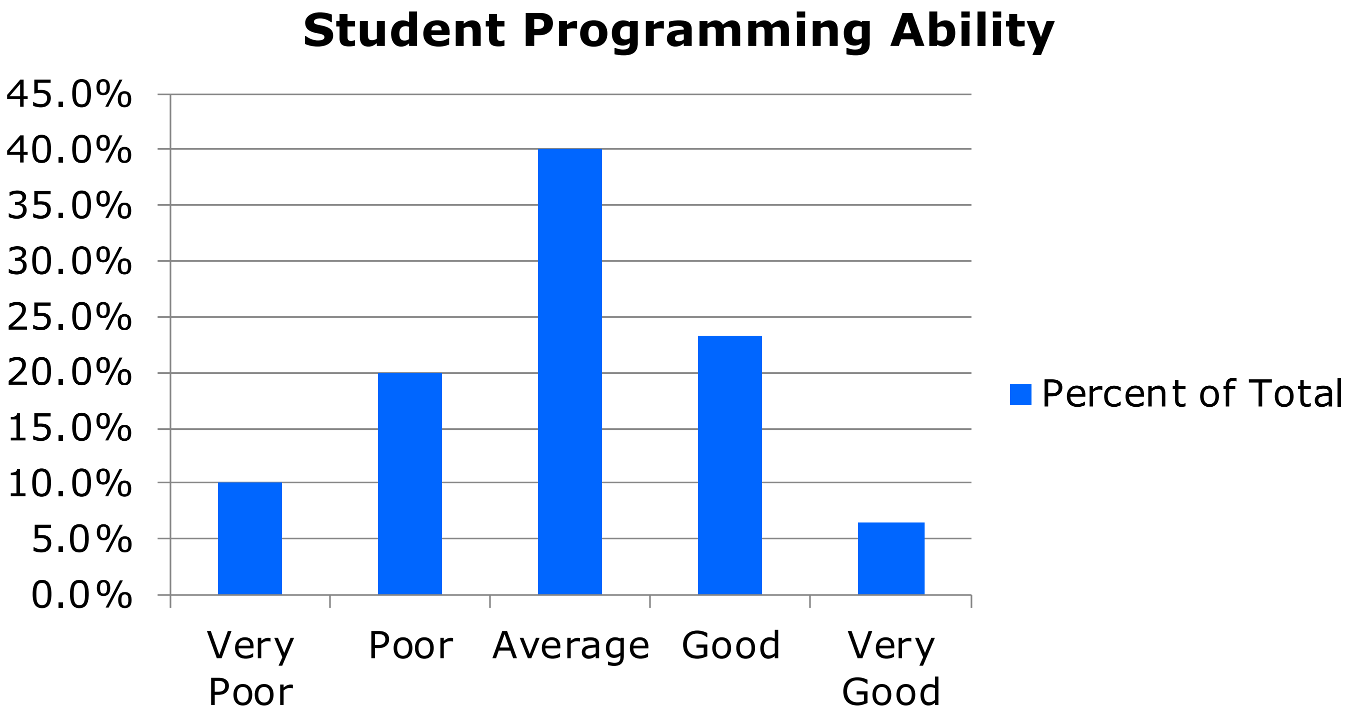
Distributions

Suppose you evaluate 30 students on their programming ability and come up with the following values:

Overall Ability	**Total**	**Percent**
Very Poor	3	10.0%
Poor	6	20.0%
Average	12	40.0%
Good	7	23.3%
Very Good	2	6.6%
Total	30	100%

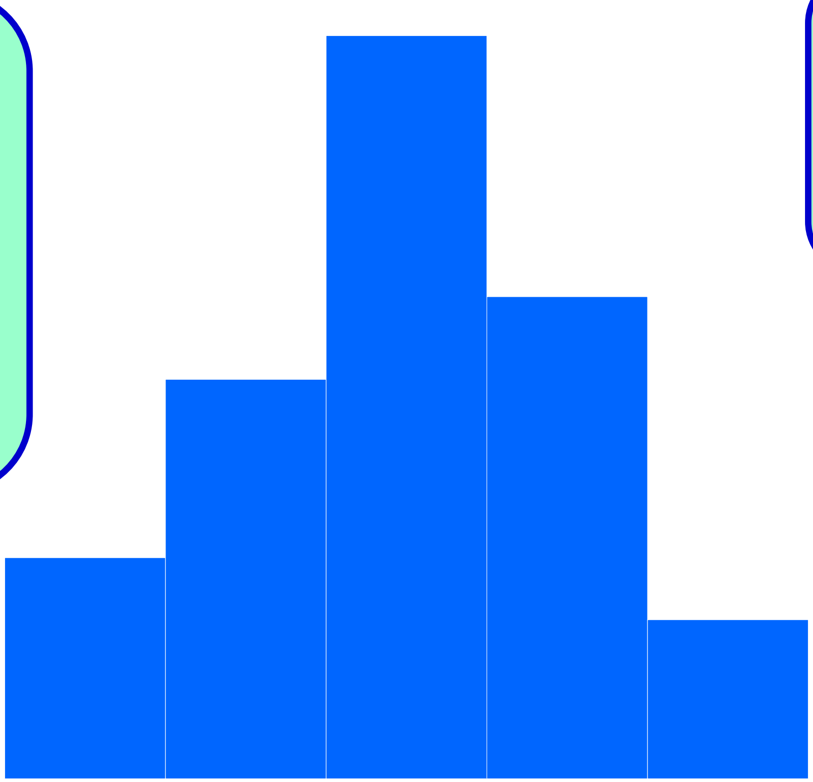
Since all students are accounted for, this is called a *distribution* of student ability values.

The Distribution as a Bar Graph



The Distribution as a Histogram

A histogram looks like a bar chart with no spaces between the bars.



Histograms are often used to show distributions.

Probability Distributions

Suppose one student is selected at random from the group of 30.

What is the probability that the student will have any given ability level?

Overall Ability	**Total**	**Probability**
Very Poor	3	.1
Poor	6	.2
Average	12	.4
Good	7	.233
Very Good	2	.066
Total	30	1.0

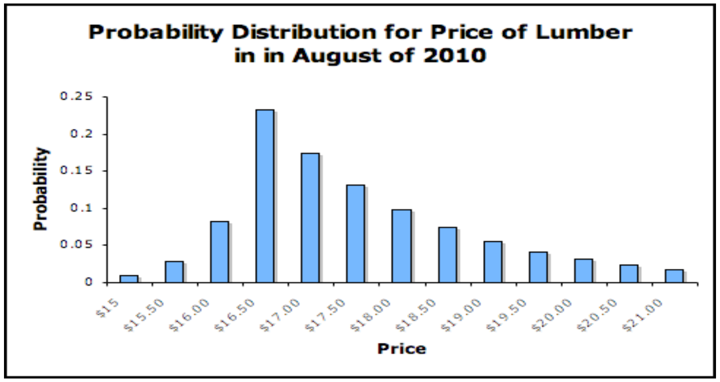
If you represent the percentage as a probability, this is called a *probability distribution* of student ability values.

Requirements for a Valid Probability Distribution

A probability distribution is the *assignment of probability values* to *each of the possible outcomes*

- **The probabilities must be numbers between 0 and 1**
- **The probabilities must add up to 1**

The probability distribution represents the *likelihood that a randomly chosen value will have a given outcome*.



Probability Distribution for Price of Lumber in in August of 2010

Calculating Probability of Other Situations

Suppose you want to know if a randomly chosen student will be *at least average* in ability.

Overall Ability	Total	Probability
Very Poor	3	.10
Poor	6	.20
Average	12	.40
Good	7	.233
Very Good	2	.066
Total	30	1.0

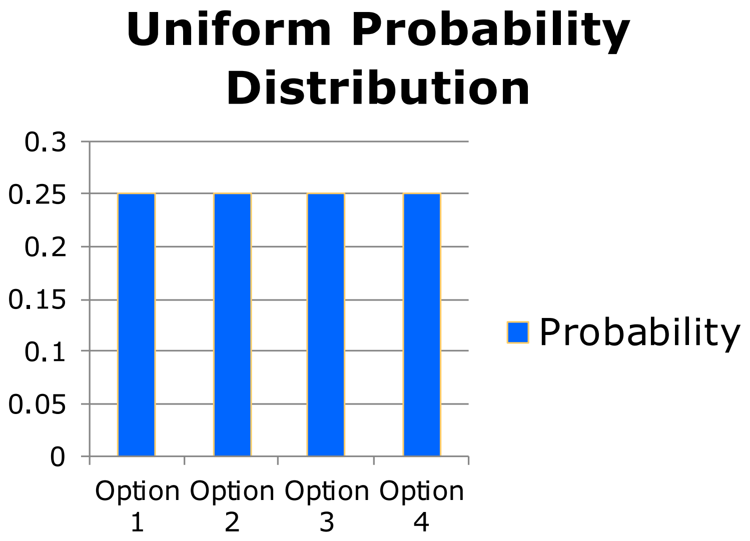
Add up the probabilities:
.40 + .233 + .066 = .70

These are average or higher

The probability of a student being at least average is .70

Uniform Distribution

- **If all events are equally likely, this is called a *uniform distribution*.**
- **If there are N options, the probability of any of them is 1/N**



Suppose There are Many Possible Values

Example: The ages of the software development staff

- Total staff size is 200 people
- Range of ages is from 18 to 72

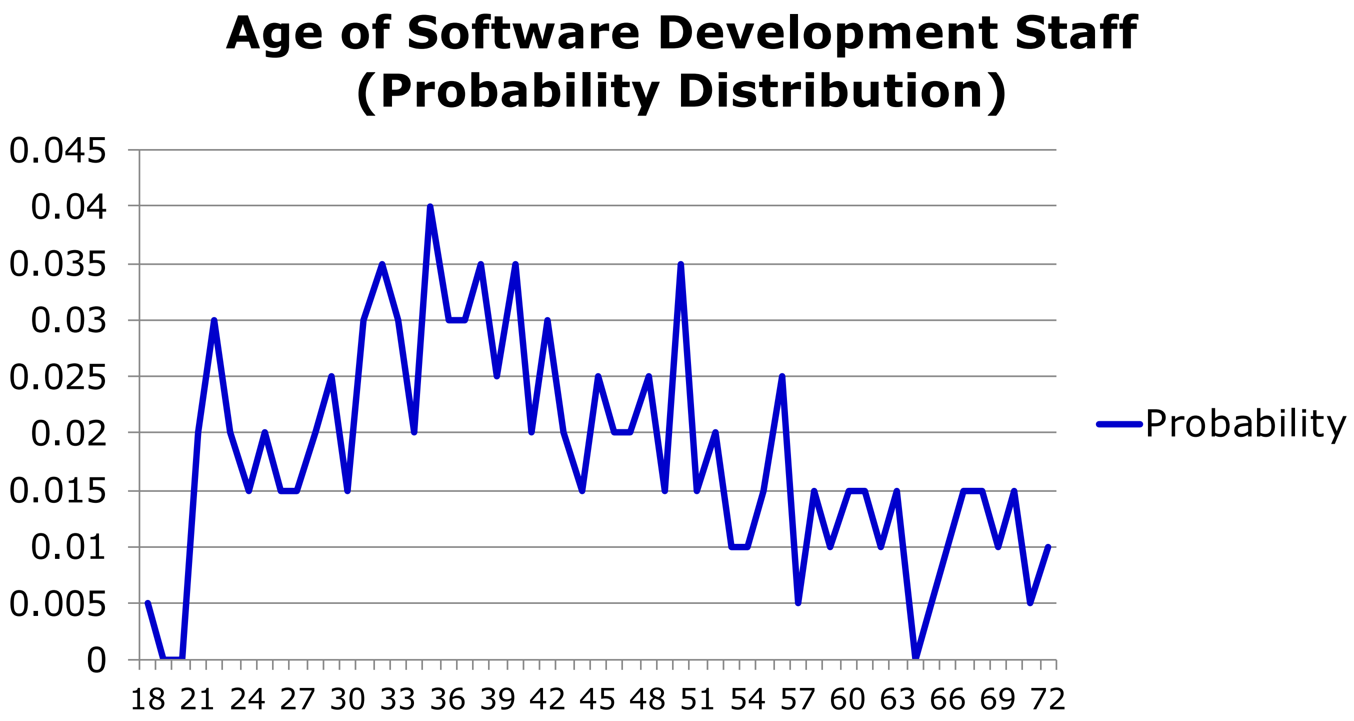
There are 55 possible values

Age	Total	Probability
18	1	.005
19	0	.000
20	2	.010
21	0	.000
22	1	.005
…	…	…
Total	200	1.0

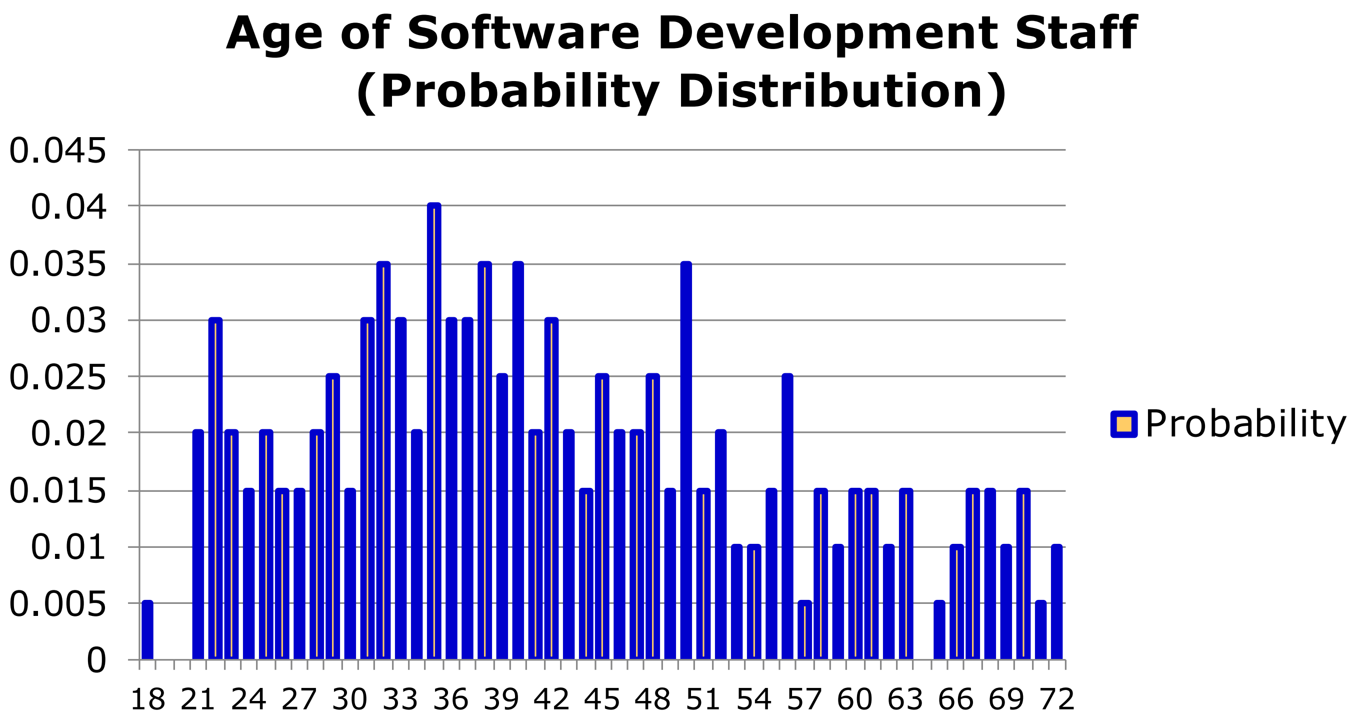
There would be a *very long list* and *very small probabilities* for most cases.

This might be very cumbersome to analyze.

Graph of Probability Distribution – Line Chart



Graph of Probability Distribution – Bar Chart



What If There Are an Infinite Number (or a very large number) of Possible Values

Examples:

- Number of lines of code in each software product.
  - Range: 11 to 128,045

In this case, it could be *meaningless to have a separate probability* for each possible value.

Solution 1: *Discretization*

- Divide the possibilities into discrete ranges that make sense for your purposes

Solution 2: *Continuous Function*

- Represent the values with a continuous function

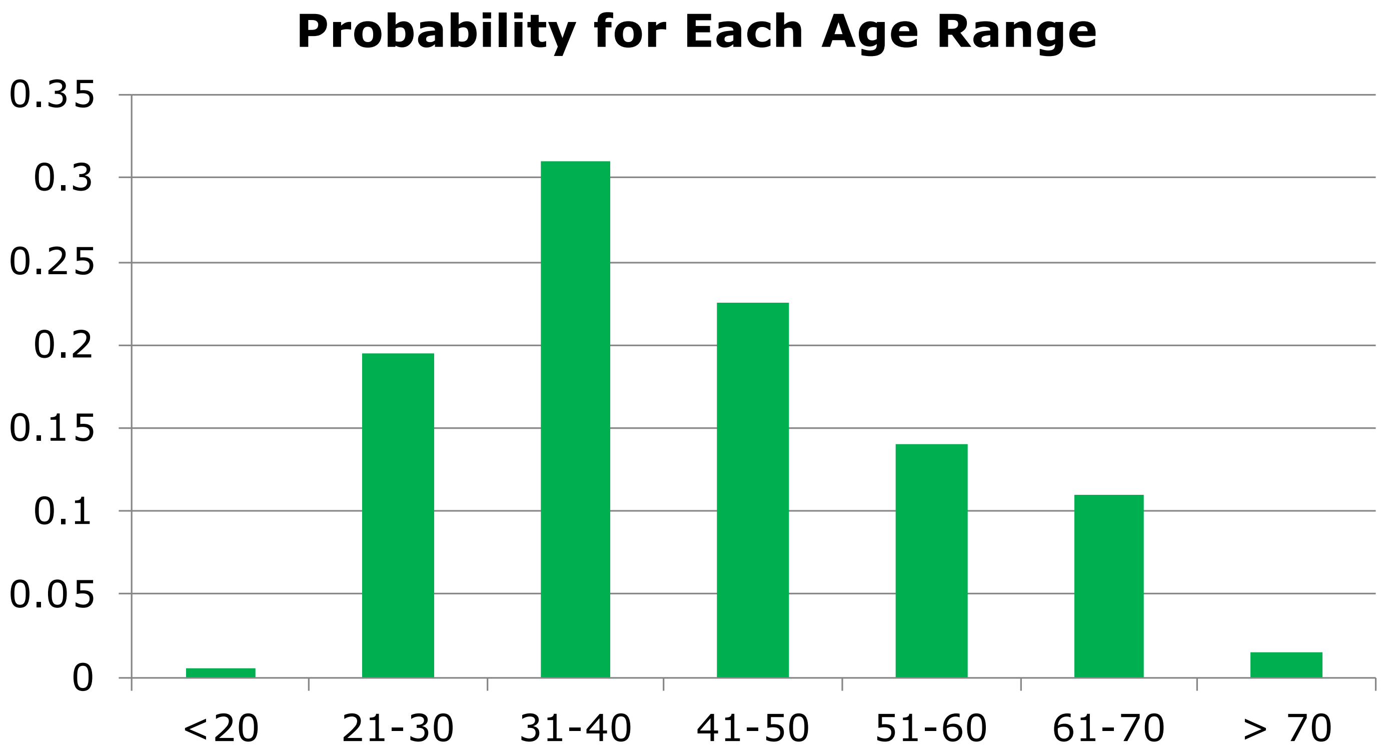
Discretization

Divide the list into discrete intervals

Age Range	Total	Probability
20 or less	1	.005
21-30	39	.195
31-40	62	.310
41-50	45	.225
51-60	28	.140
61-70	22	.110
Over 70	3	.015
Total	200	1.0

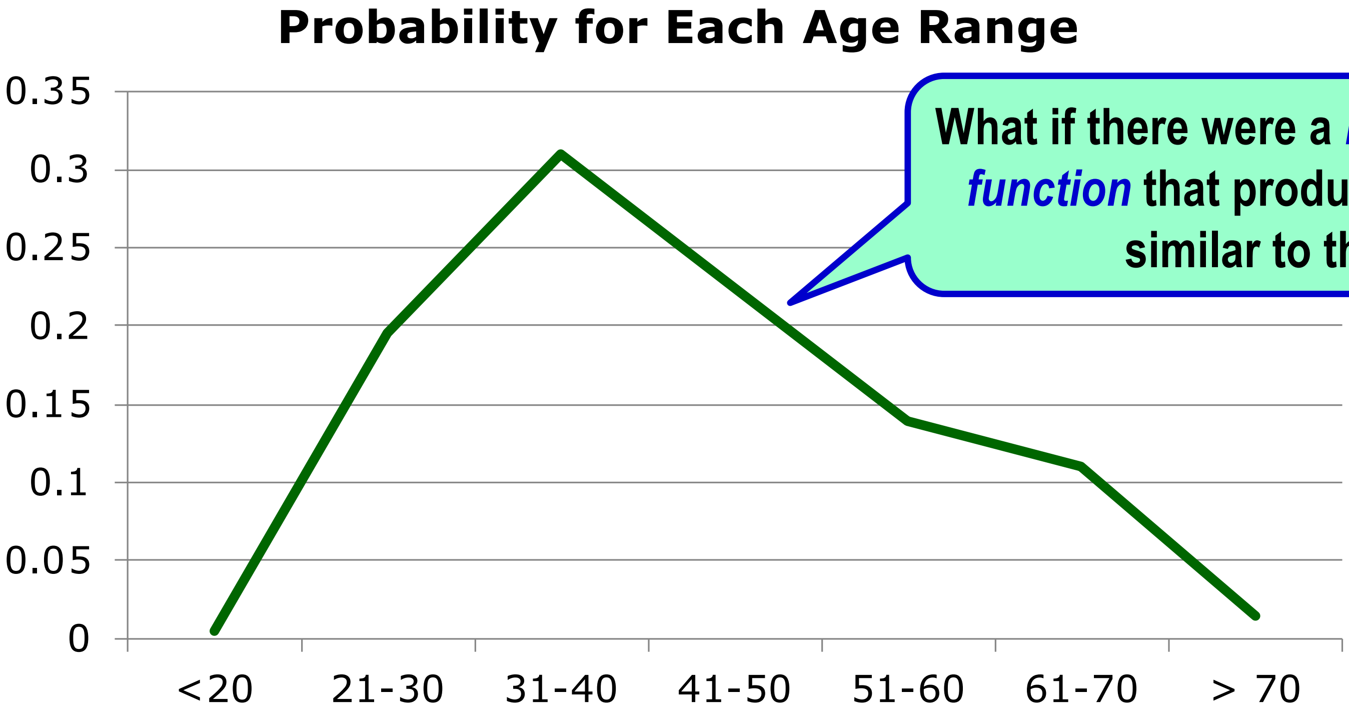
For many purposes, these categories would be useful and having only 7 categories makes analysis a lot easier than having 55 of them.

Bar Graph of Discretized Distribution Data for Programmer Age



Note that this discretized graph tells us things that were harder to see before.

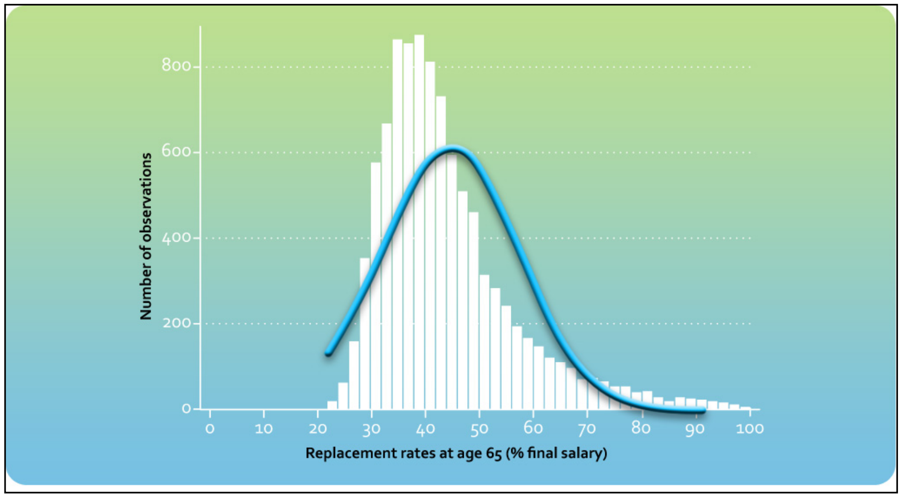
Same Graph Using a Line Chart



What if there were a ***mathematical function*** that produced a curve similar to this?

Use of a Continuous Distribution Function

For very large numbers (or infinite possibilities), it is sometimes convenient to use a *continuous function* whose shape approximates the distribution of the data.



The area under the curve is equal to 1.0 – the sum of all probabilities.

Advantages and Drawbacks of Continuous Functions

Advantages

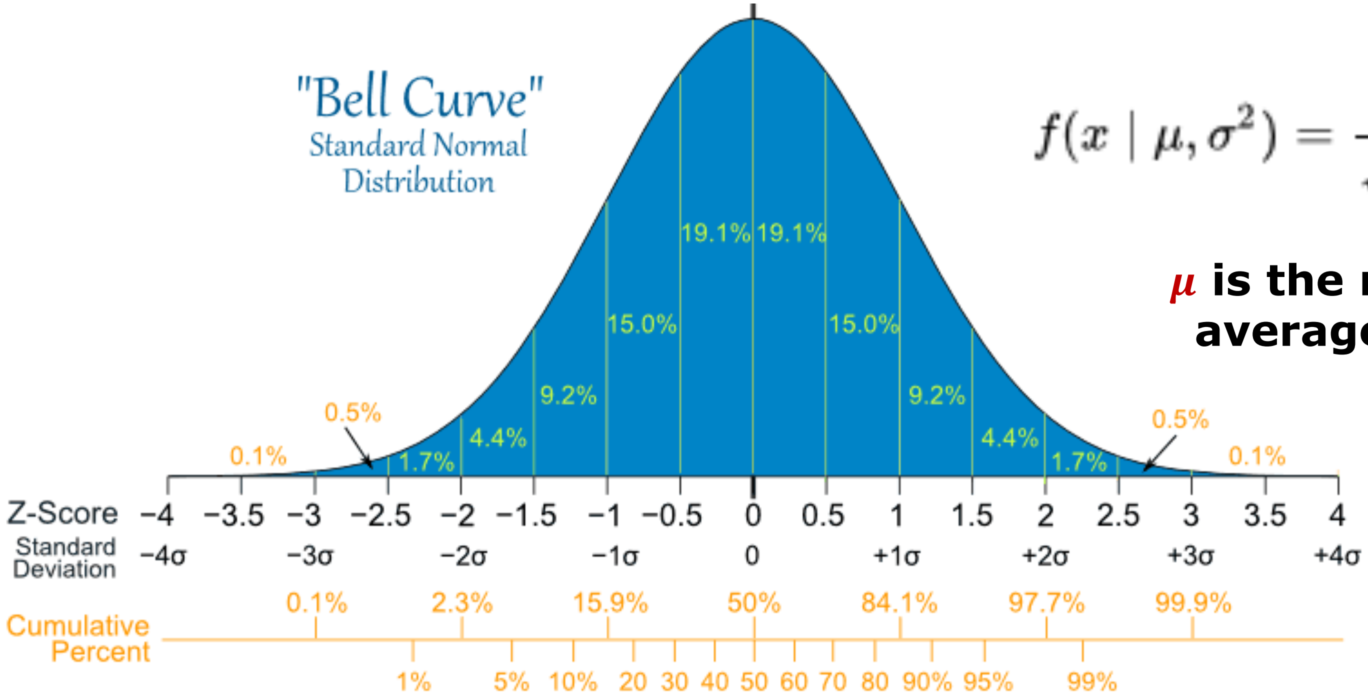
- ***Many mathematical and statistical analyses*** can be performed on the functions
  - You can often make good predictions
  - Or combine multiple independent variables and determine their relationships

Drawbacks

- You ***cannot draw conclusions*** about ***individual values***
  - For example: what is the probability that a programmer is exactly 34.25 years old?
- You must draw conclusions about ***ranges of values***
  - For example: what is the probability that a programmer is between 34 and 35 years old?

Normal Distribution (Bell Curve)

This is a widely used continuous distribution because many phenomena fit this shape.

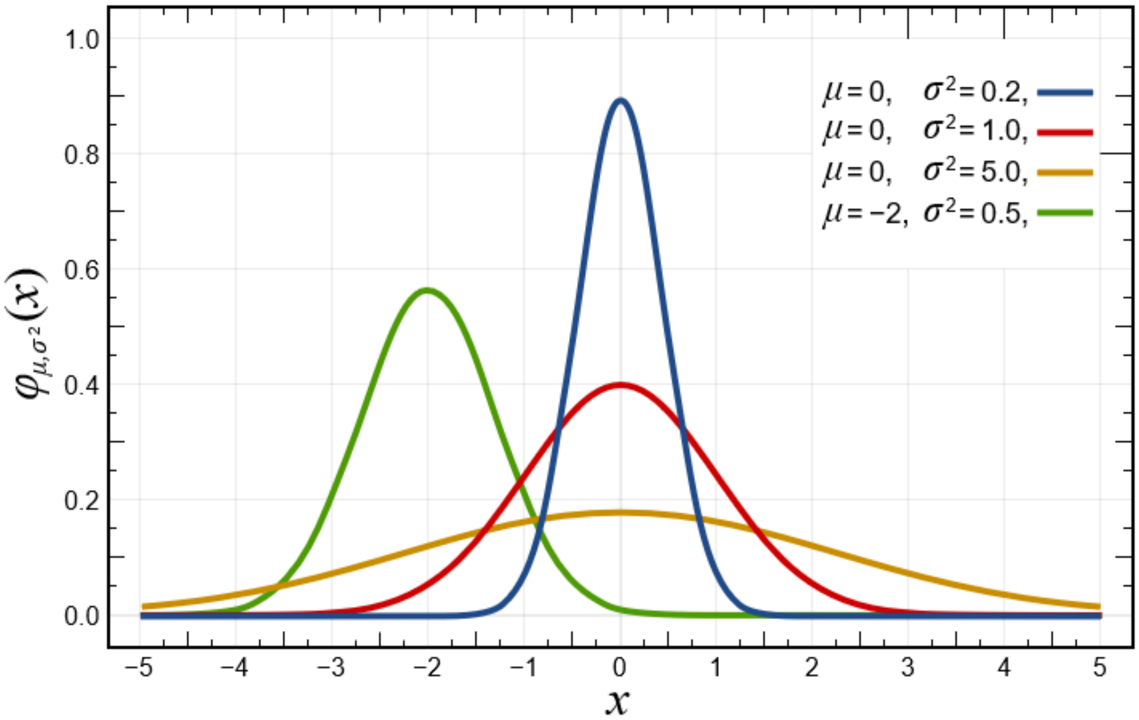


$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

μ is the mean or average value

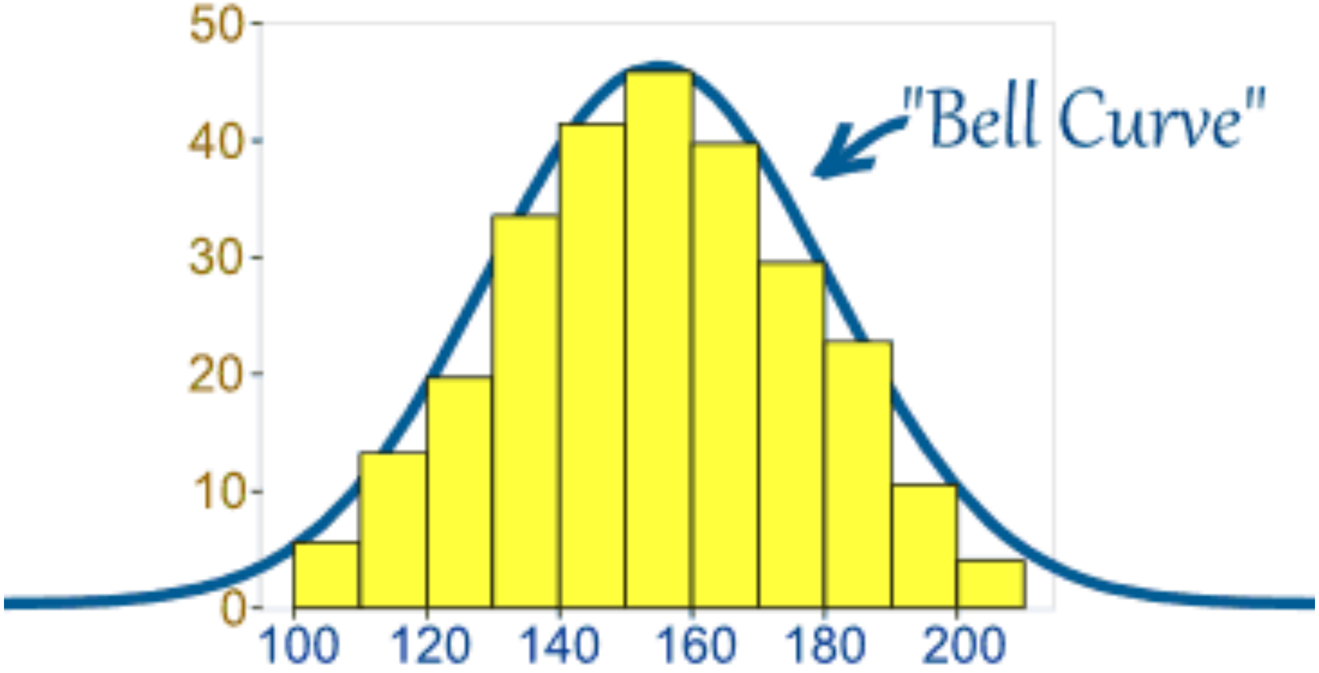
Normal Distribution Variations

$$f(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

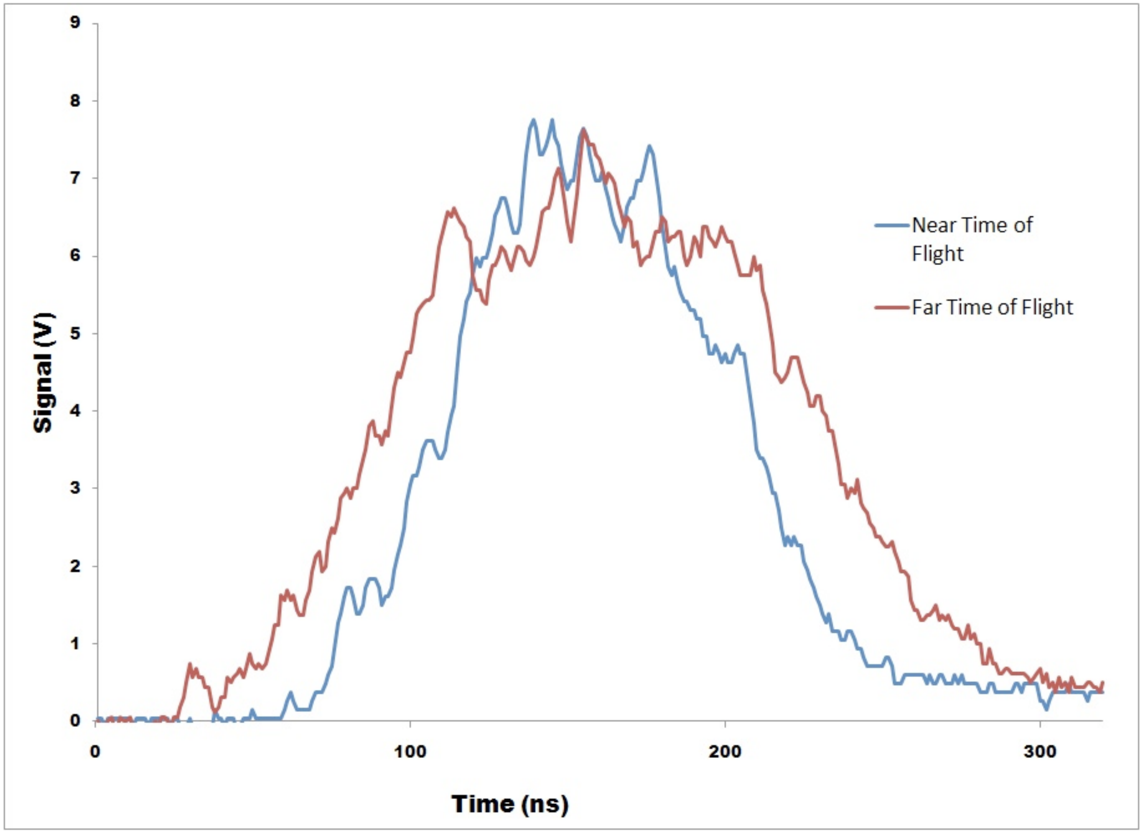


By Inductiveload - self-made, Mathematica, Inkscape, Public Domain, https://commons.wikimedia.org/w/index.php?curid=3817954

Example
Bell Curve Approximates Data



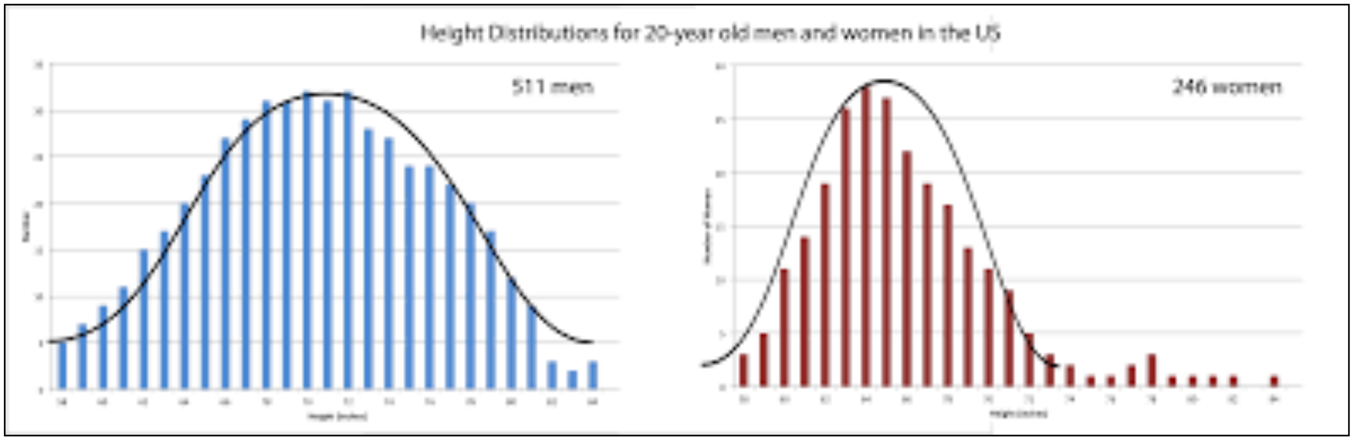
Real Data With Approximately Normal Distribution



Uses of Normal Distribution (1 of 2)

If the data are distributed in a pattern that is similar to normal distribution, *we can often reach conclusions about the data* from known facts about the normal distribution

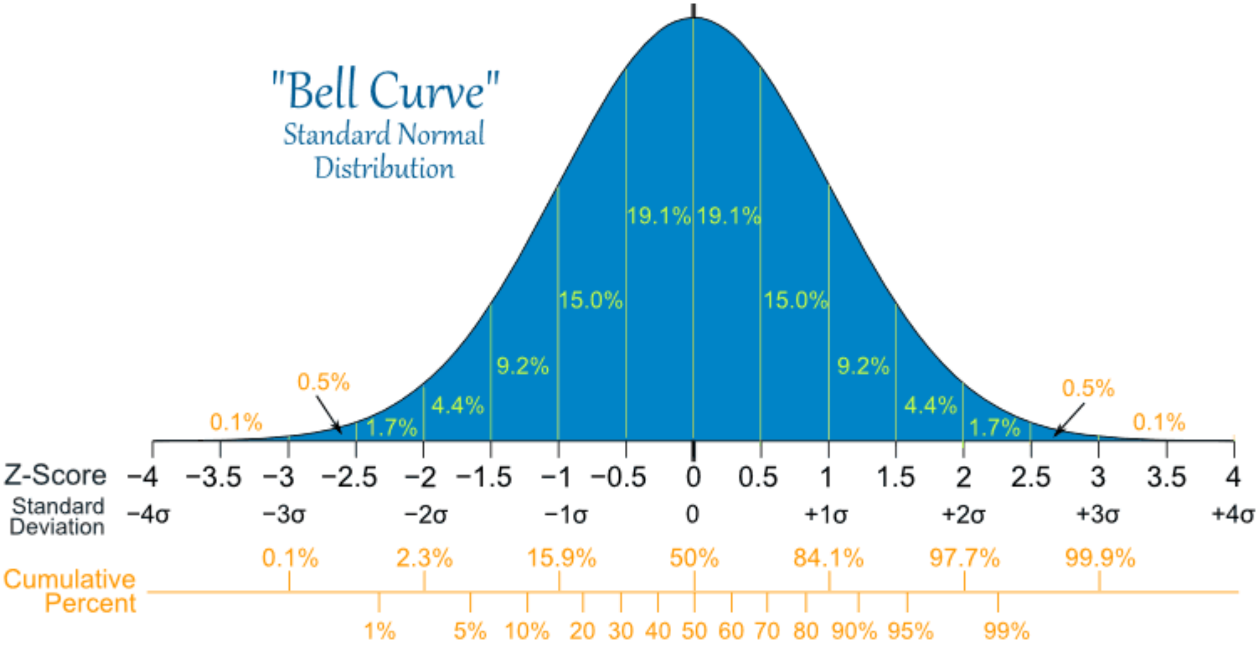
- This happens often with many natural phenomena
- For example: height



Uses of Normal Distribution (2 of 2)

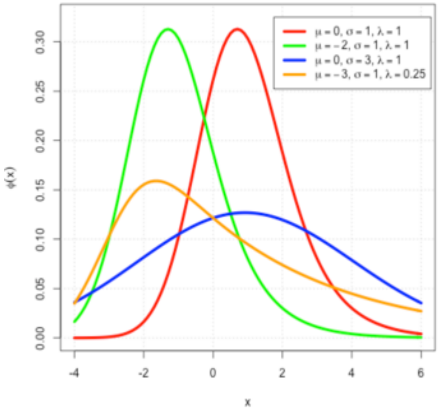
Many statistical concepts are easy to explain using the normal distribution

– Because it has many very convenient statistical and mathematical properties



Normal Distribution Variants

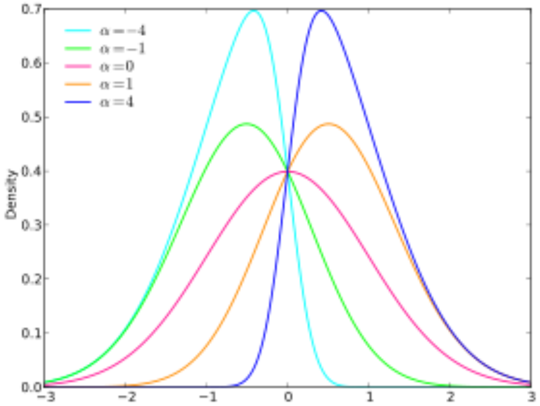
The Normal Distribution is a type of Gaussian Distribution



Various Gaussian Distributions

There are many ways to modify a Gaussian distribution to fit specific datasets

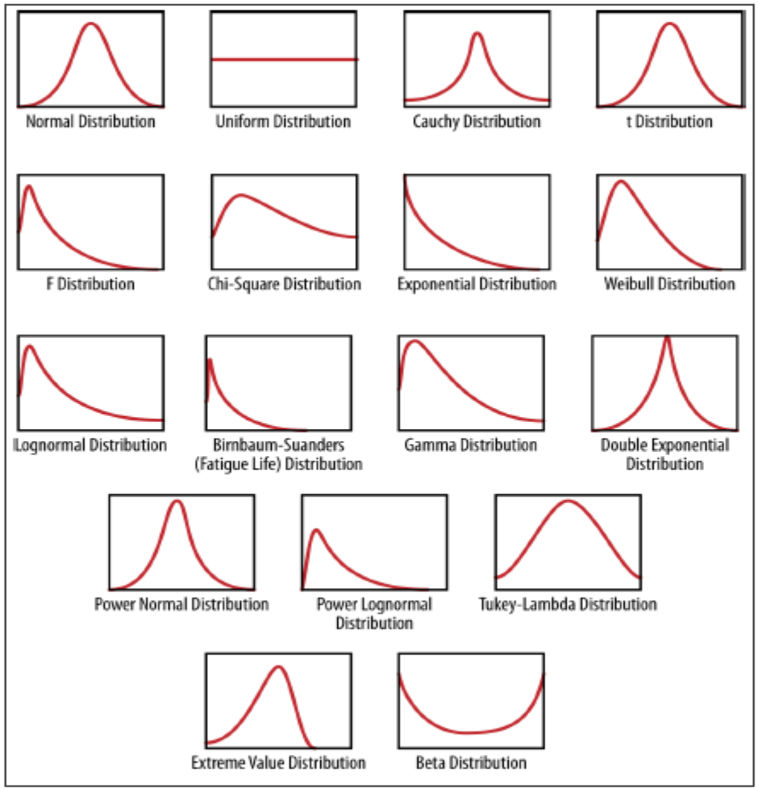
The Gaussian Distribution is a type of Elliptical Distribution



Various Elliptical Distributions

For some situations, an Elliptical distribution will fit the data

Other Examples of Continuous Distributions

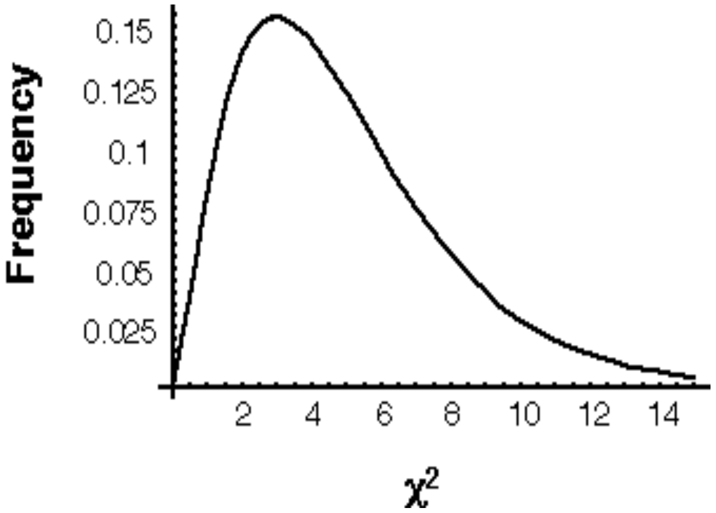


For many phenomena there are continuous distributions that approximate the actual distributions.

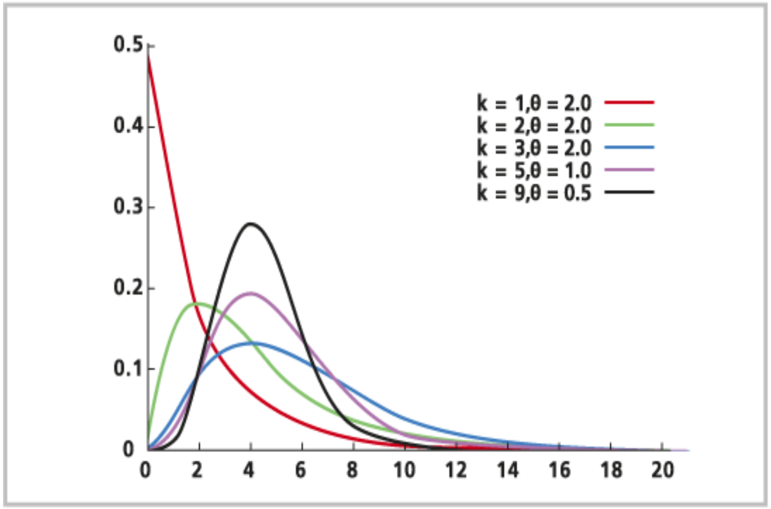
⬇

You find out using curve fitting techniques

Many Natural Phenomena Fit the Chi Square Distribution (C2 or ⍰^{2})



C2 Distribution



Variations on C2

Unlike the normal distribution, C2 is constrained to have no negative values.

Contents

- Definitions
- Scales
- Basic Analysis Approaches
- Statistical Distributions
- **Other Statistical Concepts**

Central Tendency Measures

A ***Measure of Central Tendency***

– is a

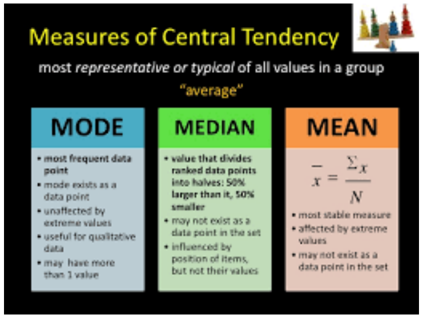
single value

– that

attempts to ***describe*** a set of data

– by

identifying the ***central position*** within that set of data.



Measures of Central Tendency

most *representative or typical* of all values in a group

"average"

MODE

- most frequent data point
- mode exists as a data point
- unaffected by extreme values
- useful for qualitative data
- may have more than 1 value

MEDIAN

- value that divides ranked data points into halves: 50% larger than it, 50% smaller
- may not exist as a data point in the set
- influenced by position of items, but not their values

MEAN

$$\bar{x} = \frac{\sum x}{N}$$

- most stable measure
- affected by extreme values
- may not exist as a data point in the set

Slideshare.net

These are also sometimes called
summary statistics
or
measures of central location

Mode, Median and Mean

(Central Tendency Measures)

- ***Mode***: **The value that appears** **most often**
- ***Median***: **The value of the** **middle** **item**
- ***Mean (*⍰ or ⍰*)***: **The** **average** **of all values**

Given these values: 1,2,2,3,4,7,9			
Term	**Description**	**Example**	**Result**
Mean	Sum / Total Number of Values	(1+2+2+3+4+7+9)/7	**4**
Median	Middle value	1,2,2,**3**,4,7,9	**3**
Mode	Most Frequent Value	1,**2**,**2**,3,4,7,9	**2**

Source: Wikipedia

Mode, Median and Mean for Various Distributions

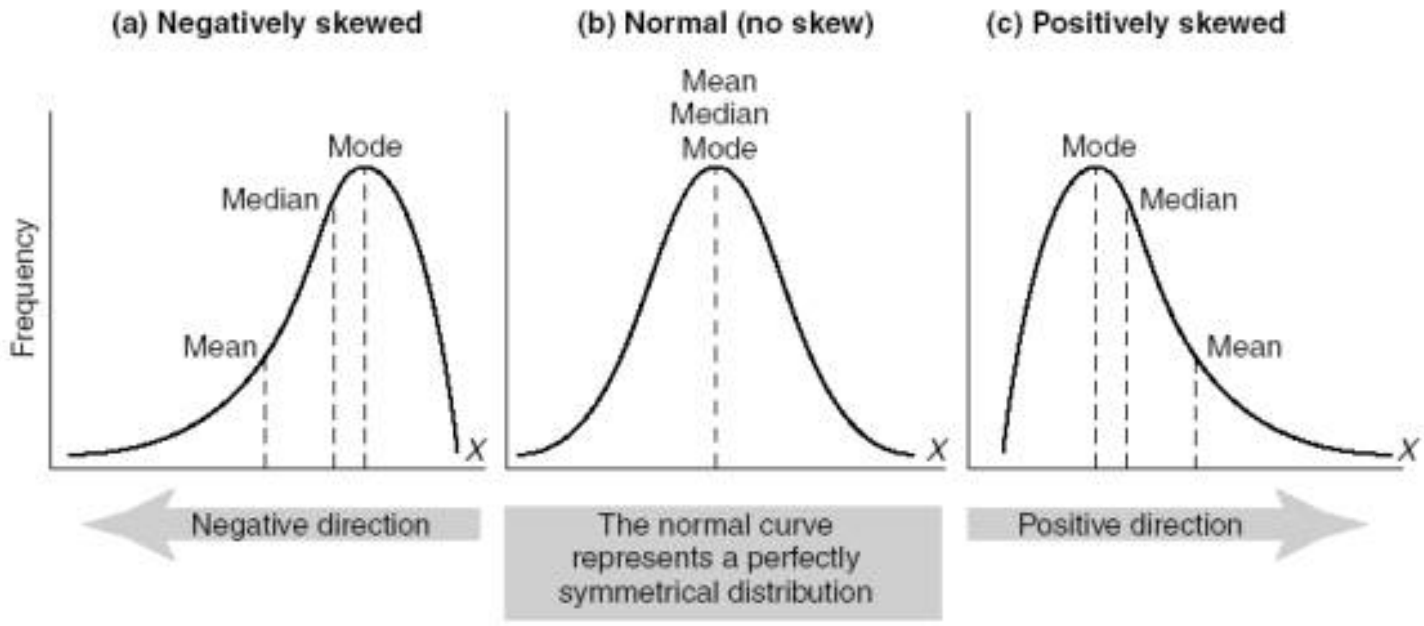
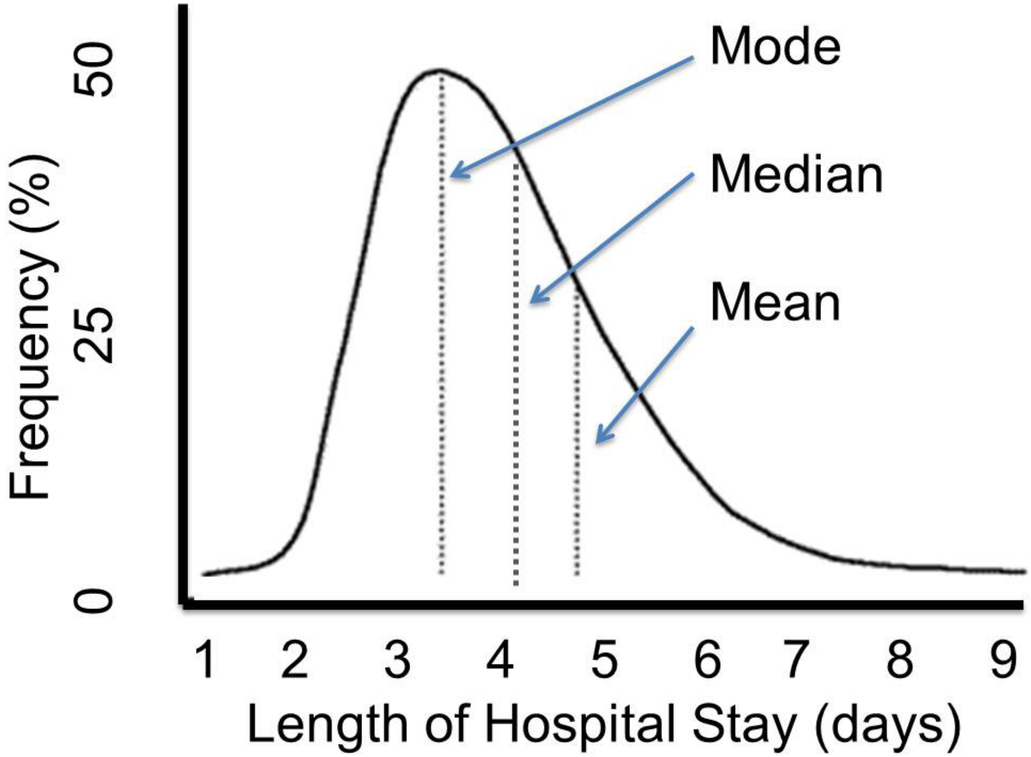


FIGURE 15.6 Examples of normal and skewed distributions

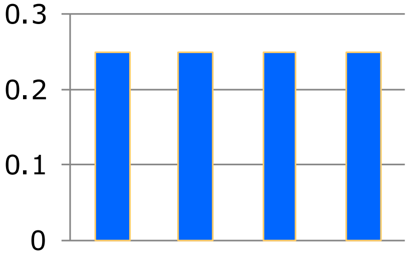
Example of Skewed Distribution

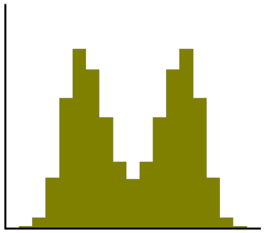


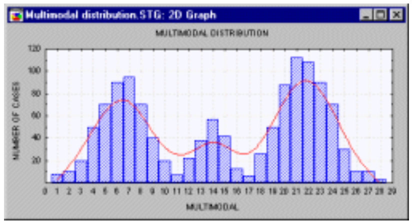
Notes about Mode (1 of 2)

1. **There can be no mode or more than one mode**

– no mode   - Bimodal   - Multimodal





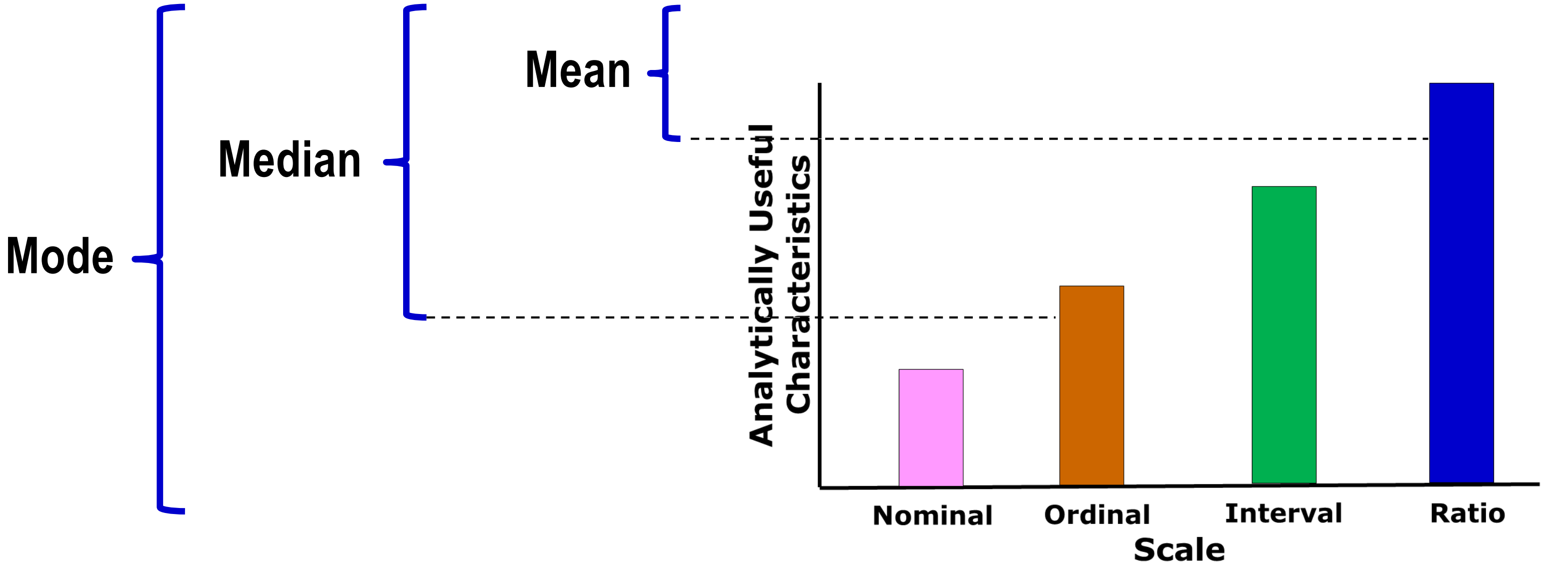


- **When there is a lot of data and there are several local maximum points, each is considered a mode**
  - i.e., *any relative high point* or peak in the data may be considered a mode.

Notes about Mode (2 of 2)

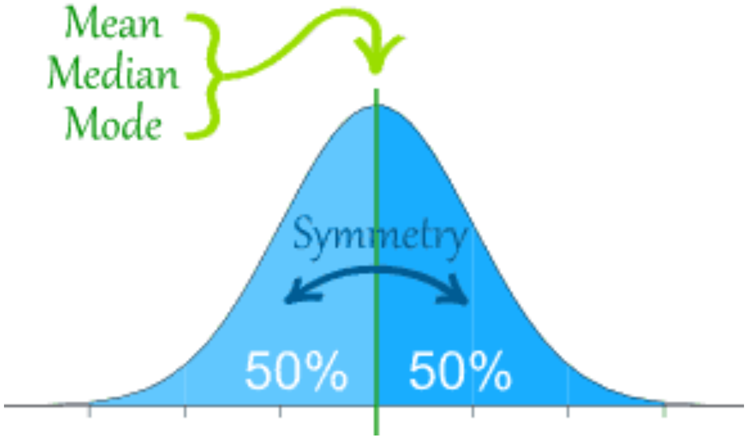
2. **The mode can be determined for data using nominal, ordinal, interval, ratio and absolute scales**

   - ***Mean & median require higher scales***



Notes about
Mode, Median and Mean

For normally distributed data, these three are equal

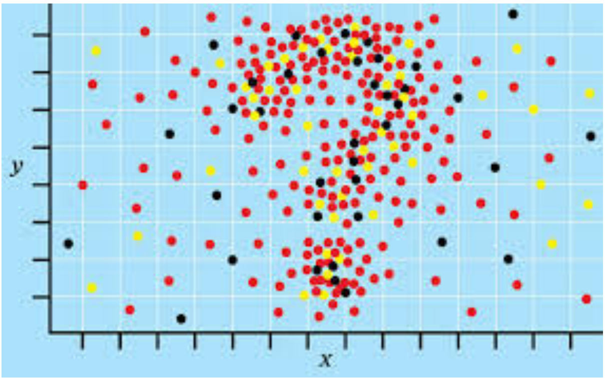


For other distributions they may not be.

- ➢ So one way to tell if the normal distribution is a possible fit to your data is to compute the mode, median and mean and see if they are the same.

Measures of Dispersion

- A ***Measure of Dispersion***
  - is a
- ***single value***
  - that
- attempts to ***describe*** a set of data
  - by
- identifying the ***spread of values*** within that set of data.

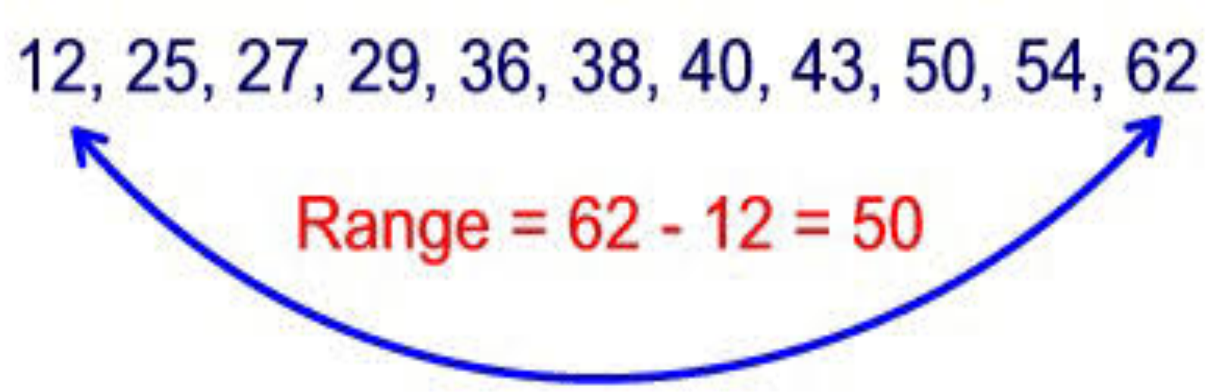


Ghhomeworktcif.gloriajohnson.us

The most common measures of dispersion are:
Range
Variance, and
Standard deviation

The Range

- **The *Range* is the *distance* between the *largest* and the *smallest* values in a set of data**
  - (Requires an interval scale)



12, 25, 27, 29, 36, 38, 40, 43, 50, 54, 62

Range = 62 - 12 = 50

Web.cs.wpi.edu

The Variance

(requires a ratio scale)

In statistics, particularly when looking at probability distributions,

the *Variance* is a measure of *how far a set of numbers is spread out*.

- This is also known as the ***dispersion*** or ***variation***.

Variance = 0

means all the numbers are the same

Variance = a small number

means all the numbers are close to each other

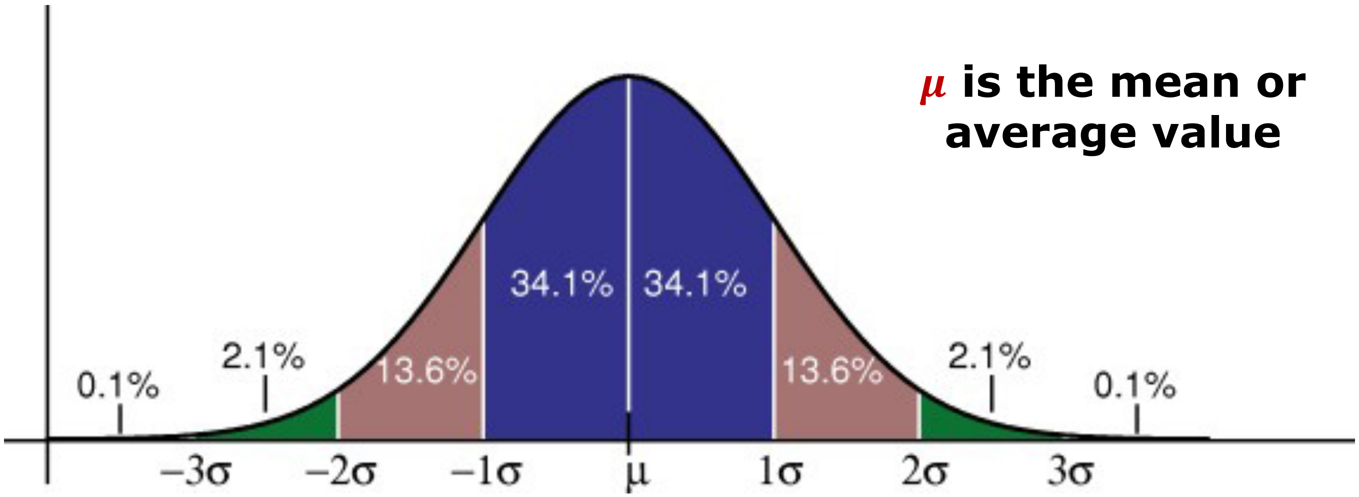
Variance = a large number

Means the numbers are widely dispersed

The Standard Deviation (⍰)

(square root of the variance)

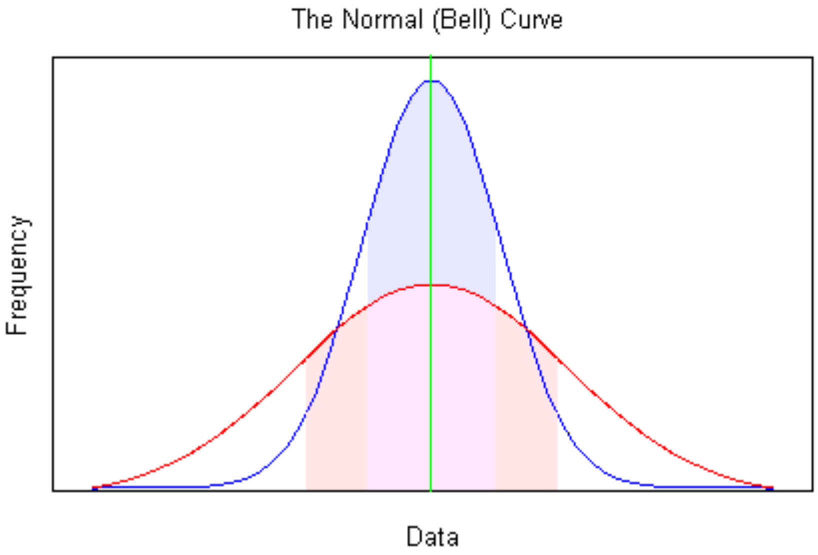
The Standard Deviation (denoted by the symbol *σ* or "sigma") is another measure used to represent the *amount of variation* or *dispersion* of a set of data values



μ **is the mean or average value**

Standard Deviation & Variance

- **If the standard deviation or variance is small, it means the data values are very close together (*blue curve*)**
- **If the standard deviation or variance is large, it means the data values are dispersed (*red curve*)**



Calculations for a Discrete, Random Variable x

Suppose you have an ordered collection of numbers, X, on a ratio scale

And suppose x is a random member of the collection

μ = mean or average value

x_i = ith possible value

p_i = probability that x has the value x_i

$Variance$ = $\sum_{i=1}^{n} p_i \ast (x_i - \mu)^2$

$\sigma = \sqrt{Variance}$

There are many uses for these calculations when doing statistical analysis of measured data

Alternative Calculations for Variance

Suppose you have an <u>ordered</u> collection of numbers, X, on a ratio scale

And suppose N is the number of members (the size) of the collection

μ = **mean or average value**

x_i = **ith value**

$$Variance = \frac{\sum (x_i - \mu)^2}{N}$$

$$\sigma = \sqrt{Variance}$$

For some data sets, this formula for variance may experience overflow, underflow or other calculation instabilities due to the sums of squares required. See "Algorithms for calculating variance" on Wikipedia.

Any Questions?

End of Lecture

References Part 1

Bourque, P. and R.E. Fairley, eds., *Guide to the Software Engineering Body of Knowledge, Version 3.0*, IEEE Computer Society Press, 2014. ISBN 978-0769551661. Available in PDF format (free) at www.swebok.org.

Crosby, Philip, *Quality is Free*. New York: McGraw-Hill, 1979. ISBN 0-07-014512-1.

Fenton, Norman and James Bieman, *Software Metrics: A Rigorous and Practical Approach, Third Edition*, Chapman and Hall, 2014. ISBN 978-1439838228.

Juran, Joseph M., *Juran on Quality by Design: The New Steps for Planning Quality into Goods and Services*. Free Press, 1992. **ISBN-13:** 978-0029166833.

Project Management Institute, *SWX – The Software Extension to the PMBOK Guide Fifth Edition, Project Management Institute, 2013. ISBN 978-1628250138.*

Weinberg, Gerald M., *Quality Software Management, Volume 1, Systems Thinking*, Dorset House, New York, 1992. ISBN: 0-932633-22-6.

References
Part 2

Devore, Jay, N. Farnum, and J. Doi, *Applied Statistics for Engineers and Scientists, 3nd Edition*, Thompson, 2013. ISBN 978-1133111368.

Fenton, Norman and James Bieman, *Software Metrics: A Rigorous and Practical Approach, Third Edition*, Chapman and Hall, 2014. ISBN 978-1439838228.

Stevens, S. S., *"On the Theory of Scales of Measurement". Science* (7 June 1946). 103 (2684): 677–680.