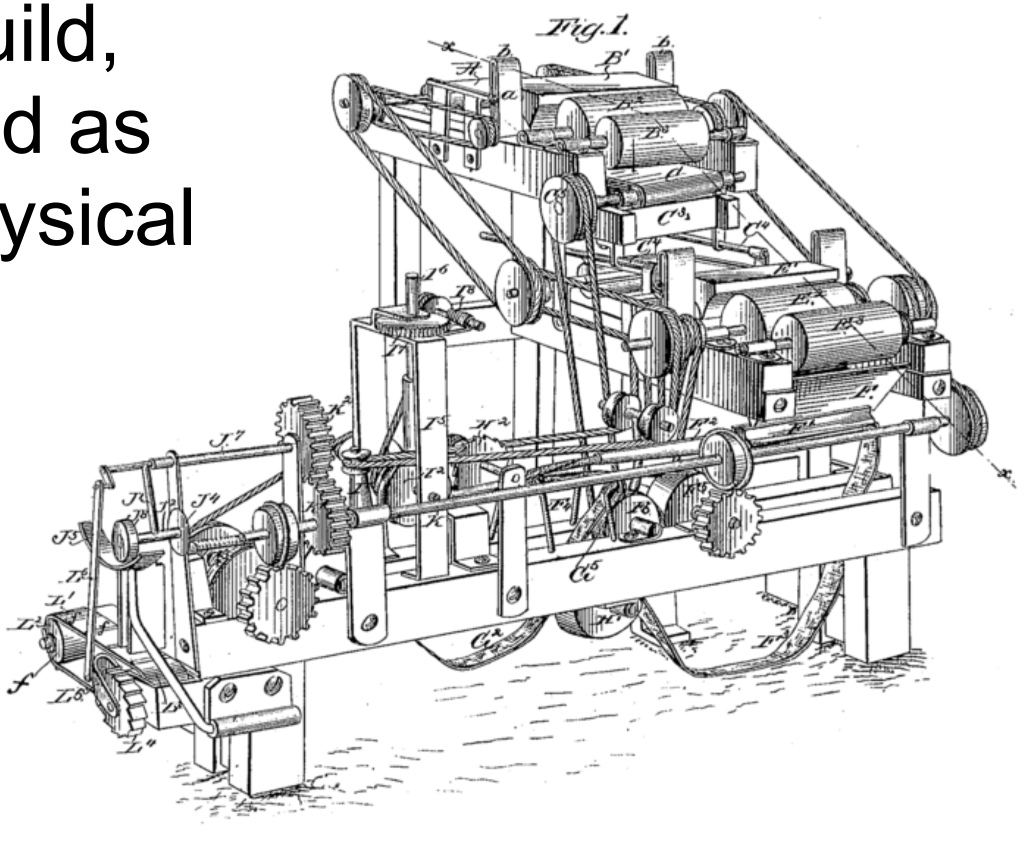
Best practices for scientific computing

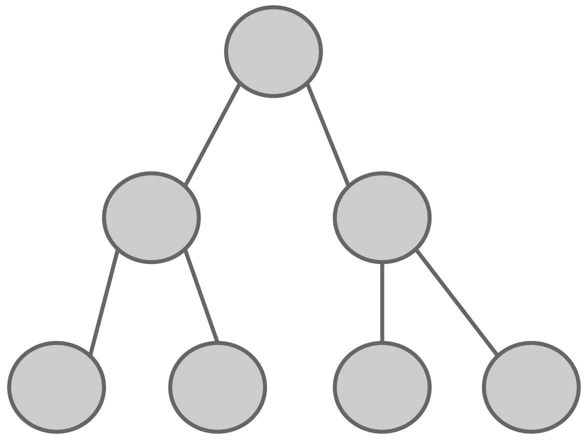
MolMed TechTalk

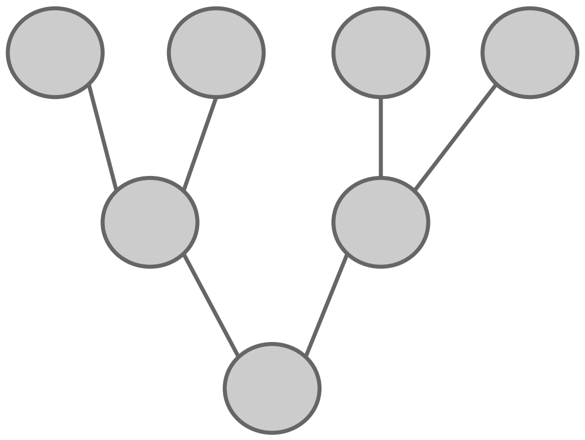
Wilson, Greg et al. "Best practices for scientific computing." *arXiv preprint arXiv: 1210.0530* (2012).

"We believe that software is just another kind of experimental apparatus and should be build, checked, and used as carefully as any physical apparatus."



Why software engineering in biology?


High energy physics


"Omics"

1. Write programs for people, not computers

- Only a handful of facts
    - Break programs into functions
- Use meaningful, consistent and distinctive names
- Consistent style and formatting
- Create work packages of ~ 1 h.

Worst practice example

```scala
object UglyCodeExample extends App {
  def m(stringToTransform: String): String = {
     def t(string: String): List[String] = {
     def th(string: String, si: Int): List[String] = {
     if (si == 0) Nil
     else {
     val (firstString, secondString) = string.splitAt(si); val newString = secondString + firstString
     newString :: th(string, si - 1)}}
     th(string, string.length)
     }
     val list: List[String] = t(stringToTransform); val sortedList = list.sortWith((x, y) => { x <= y });
val lastColumn = sortedList.map(row => row.last).mkString
     lastColumn
  }
  println(m("^BANANA|"))
}
```

A better practice example

```
object BetterCodeExample extends App {

  def burrowsWheelersTransform(stringToTransform: String): String = {

    // Carries out the transformation on the string.
    def tranformString(string: String): List[String] = {
      // The helper functions creates a list of strings until it has iterated
      // over the full length of the original string.
      def transformStringHelper(string: String, splitIndex: Int): List[String] = {
        if (splitIndex == 0)
          Nil
        else {
          val (firstString, secondString) = string.splitAt(splitIndex)
          val newString = secondString + firstString
          newString :: newString :: transformStringHelper(string, splitIndex - 1)
        }
      }
       transformStringHelper(string, string.length)
    }

    // Create a list of the transformed strings
    // Each string can be viewed as a row in a table.
    val list: List[String] = tranformString(stringToTransform);
    // Sort them the rows alphabetically
    val sortedList = list.sortWith((x, y) => { x <= y })
    // Get the last column of the table.
    val lastColumn = sortedList.map(row => row.last).mkString
    lastColumn
  }
  println(burrowsWheelersTransform("^BANANA|"))
}
```

Use an integrated development environment (IDE)!

- Autocompletion
- Easier refactoring
- Automated code formatting
- etc ....

Better code

2. Automate repetitive tasks

- History
  - combine with other commands
  - $ history | grep "cool command I used once" | tail
- Script
  - Collect sequences of commands in files
  - "Don't copy paste"
- Build tool
  - make etc
  - Check dependencies

```
foo.o : bar.c  #if bar.c is updated after foo.o, run following:
        $(CC) -I. -I$(srcdir) $(CFLAGS) -c $< -o $@
```

3. Use the computer to record history

- Use software to track computational work
- Make it traceable!
  - Unique identifiers of raw data
  - Unique identifiers for programs and libraries
  - Record parameters used to generate output

Example

Taken from Version4a22f.txt

Unique id for run

Git commit-hash for R-scripts

Git hash for each script run

Git commit-hash for preceeding SNV-caller-run

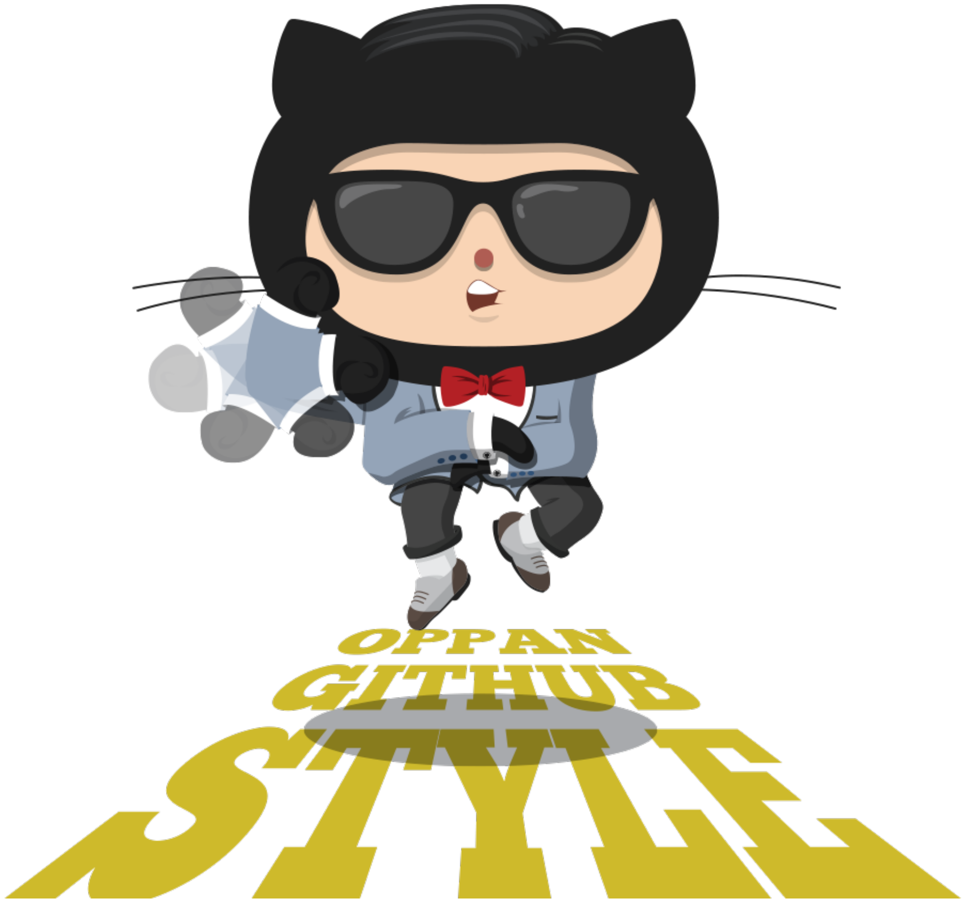
```
4a22fdb58db80108cc4af24686959193 -
Thu Apr 25 13:23:56 CEST 2013
Git commit hash: commit 487beac15edd86f22195db5a33d6cbe44cbbfc9b
VCF-files read from:/proj/b2012134/private/work/Recalibration/VariantCallingAllCandidates/Phred0MAPQ0/
........
Allele_fractions_in_pools.R has git-sha1: e4b329471061788158e9e138b3c493db876fb1af
Allele_fractions_and_sequencing_depth_of_germline_SNPs_in_HaloPlex_and_WGS_data.R has git-sha1:
dba697212a2b9abfe61af65a07c137ff815153b8
........

DATA FROM SNVCALLING:
Wed Apr 17 13:42:15 CEST 2013
/bubo/home/h12/carll/GIT/ALL/BATCH/StartScripts
Git commit hash: commit 873f5ff7ee5f791883ad213b220bbe5599aaa778
/bubo/home/h12/carll/GIT/ALL/Halo/python/BasicFixedPointSNVcaller.py has git-sha1: 61ef19e0ae8fa30d7b7e412e6d77fe6d8cd2ee2c
PhredCutoff: 0
MAPQCutoff: 0
```

4. Make incremental changes

- Strategy:
  - Industry: planning months-years in advance in accordance to specifications
  - Academia: Follow the path that last run indicated
- Implementation:
  - Iteration of modules of ~1h (valid wherever humans do the programs)
- Track in ...

5. Use version control

- Use version control for everything that is created manually



6. DRY – don’t repeat yourself

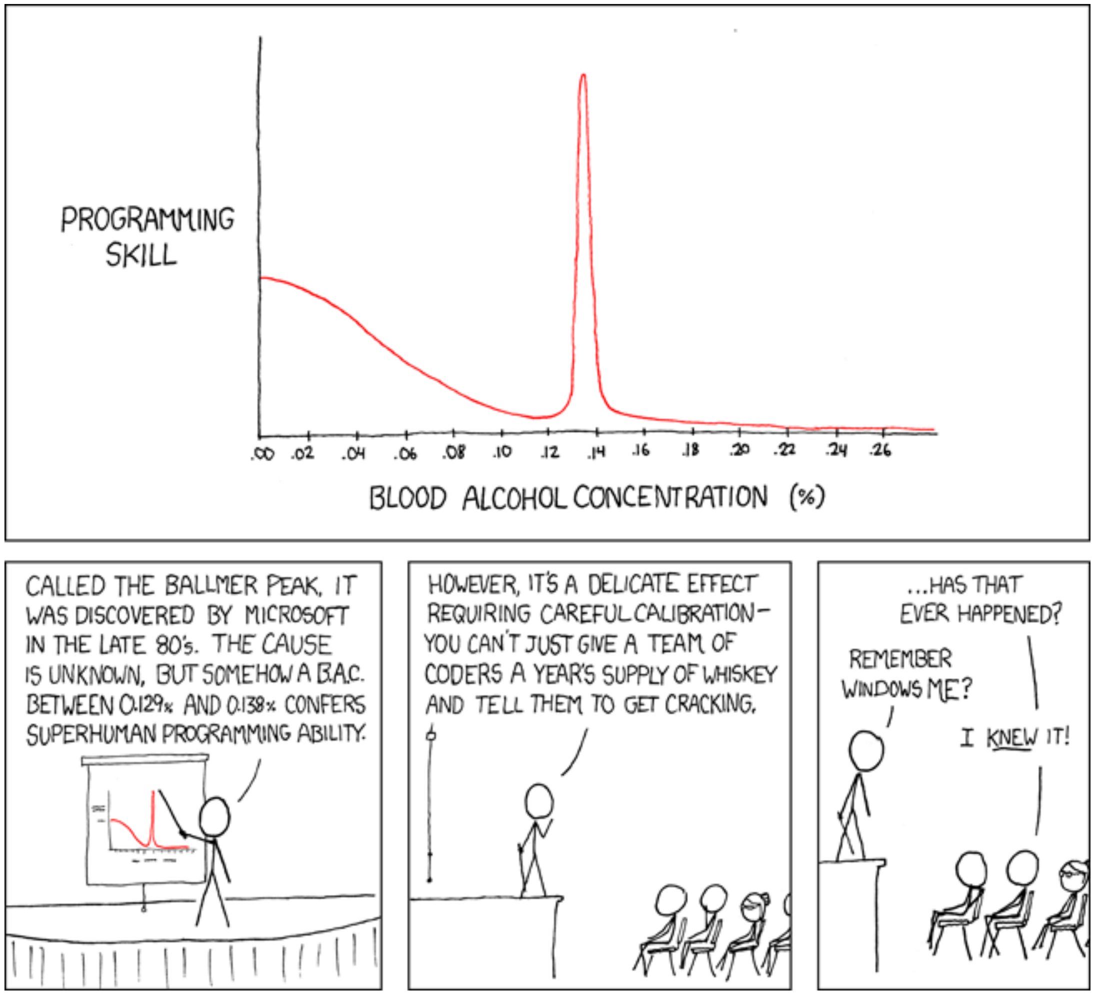
Data:

- Single representation for each piece of data ("database")

Code

- Modularize - don't copy paste around code
- Use external modules if available (≈ choose a language where others have made the most progress)

7. Plan for mistakes


PROGRAMMING SKILL
.00 .02 .04 .06 .08 .10 .12 .14 .16 .18 .20 .22 .24 .26
BLOOD ALCOHOL CONCENTRATION (%)
CALLED THE BALLMER PEAK, IT WAS DISCOVERED BY MICROSOFT IN THE LATE 80's. THE CAUSE IS UNKNOWN, BUT SOMEHOW A B.A.C. BETWEEN 0.129% AND 0.138% CONFERS SUPERHUMAN PROGRAMMING ABILITY.
HOWEVER, IT'S A DELICATE EFFECT REQUIRING CAREFUL CALIBRATION— YOU CAN'T JUST GIVE A TEAM OF CODERS A YEAR'S SUPPLY OF WHISKEY AND TELL THEM TO GET CRACKING.
...HAS THAT EVER HAPPENED?
REMEMBER WINDOWS ME?
I KNEW IT!

- Defensive programming
  - Add assertions to check that program logic holds while the program runs
  - Adds executable documentation!
- Automated testing
  - Unit tests
  - Integration tests
  - Regression testing

Defensive programing

```scala
def burrowsWheelersTransform(stringToTransform: String): String = {

    require(!stringToTransform.isEmpty(), "You are trying to transform an empty string!")

    // Carries out the transformation on the string.
    def tranformString(string: String): List[String] = {

        // The helper functions creates a list of strings until it has iterated
        // over the full length of the original string.
        def transformStringHelper(string: String, splitIndex: Int): List[String] = {

            require(splitIndex >= 0)
           [...]
        }
        transformStringHelper(string, string.length)
    }
 [...]
}
```

Automated testing

```
import org.scalatest.FunSuite

import BetterCodeExample._

class BetterCodeExampleUnitTests extends FunSuite {

  test("Burrow wheelers transform of ^BANANA|") {
     assert(BetterCodeExample.burrowsWheelersTransform("^BANANA|") === """BNN^AA|A""")
  }

  test("Burrow wheelers transform of a empty string") {
     intercept[IllegalArgumentException] {
            (BetterCodeExample.burrowsWheelersTransform(""))
     }
  }
}
```

Why do this?

Confidence that
what you're doing
is actually working

8. Optimize only when necessary

"We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil"

"Computer Programming as an Art (1974)"   Donald Knuth

- Choose the highest-level language possible (with access to suitable libraries)
- Optimize afterwards

Java:

```
public class HelloWorld
{
   public static void main (String[] args)
   {
      System.out.println("Hello, world!");
   }
}
```

Python

```
print "Hello, world!"
```

Example - ChainSum

C

```c
int chainsum(int n) {
   int result = 0;
   for (int i = 1; i <= n; ++i)
       result += i;
   return result;
}
```

Scala

Functional

```scala
def chainsum(n: Int) = if(n == 0) 0 else n + chainsum(n-1)
```

Imperative

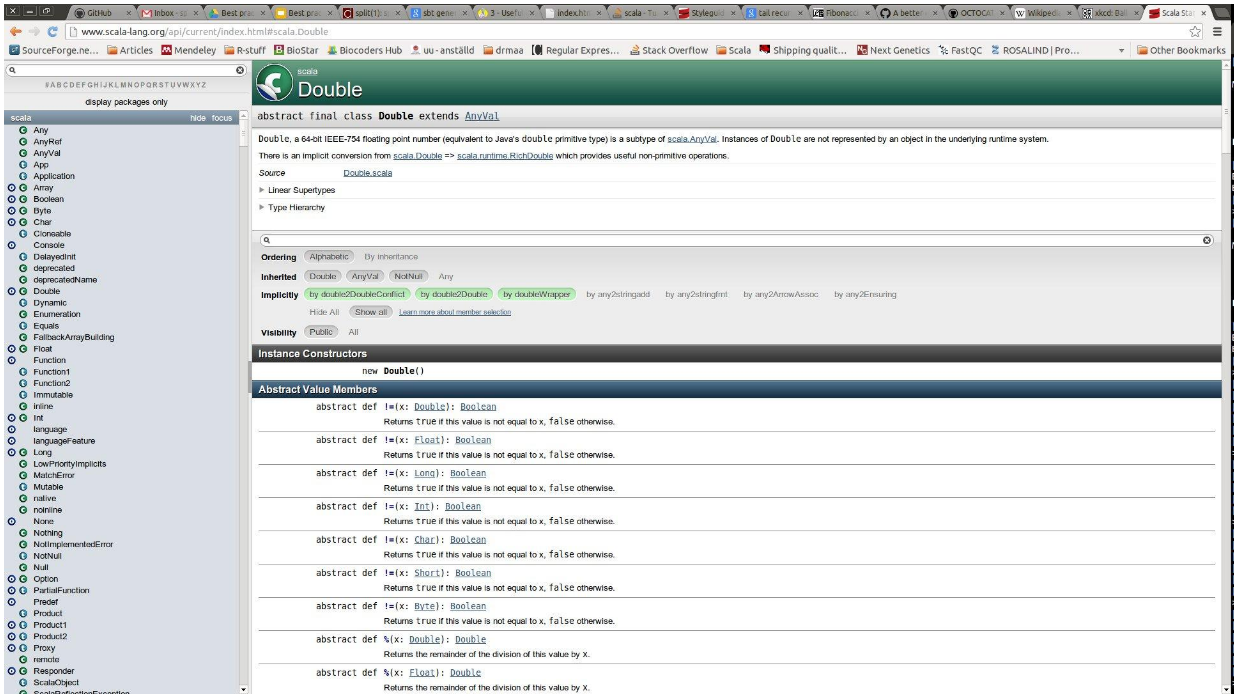
```scala
def chainsum(n: Int)={
 var res = 0
 for(i <- 1 to n)
   res +=i
 res
}
```

9. Document design and purpose, not mechanics

```
/**
    * Burrows-Wheeler transform of string
    *
    * Perform the Burrows-Wheeler transform on a string. See http://en.wikipedia.
org/wiki/Burrows%E2%80%93Wheeler_transform
    * for an introduction to the algorithm.
    *
    * This particular implementation is recursive.
    *
    * @param stringToTransform the string to transform
    * @return the burrows wheeler transform of the input string.
    */
   def burrowsWheelersTransform(stringToTransform: String): String = {[...]}
```

Use automated documentation tools

- If the documentation of the program is embedded in the program, you can change it as you change the code.



scala

Double

`abstract final class Double extends AnyVal`

`Double`, a 64-bit IEEE-754 floating point number (equivalent to Java's `double` primitive type) is a subtype of scala.AnyVal. Instances of `Double` are not represented by an object in the underlying runtime system.

There is an implicit conversion from scala.Double => scala.runtime.RichDouble which provides useful non-primitive operations.

Source Double.scala

▸ Linear Supertypes

▸ Type Hierarchy

Instance Constructors

```
new Double()
```

Abstract Value Members

```
abstract def !=(x: Double): Boolean
```
Returns `true` if this value is not equal to x, `false` otherwise.

```
abstract def !=(x: Float): Boolean
```
Returns `true` if this value is not equal to x, `false` otherwise.

```
abstract def !=(x: Long): Boolean
```
Returns `true` if this value is not equal to x, `false` otherwise.

```
abstract def !=(x: Int): Boolean
```
Returns `true` if this value is not equal to x, `false` otherwise.

```
abstract def !=(x: Char): Boolean
```
Returns `true` if this value is not equal to x, `false` otherwise.

```
abstract def !=(x: Short): Boolean
```
Returns `true` if this value is not equal to x, `false` otherwise.

```
abstract def !=(x: Byte): Boolean
```
Returns `true` if this value is not equal to x, `false` otherwise.

```
abstract def %(x: Double): Double
```
Returns the remainder of the division of this value by x.

```
abstract def %(x: Float): Double
```
Returns the remainder of the division of this value by x.

If it is not possible to understand the code without comments, you're probably doing it wrong...

10. Collaborate

Would you deliver your thesis without anyone else had read it?

Mutual Code reading once a week?

The end

All code (Johan wrote) is available here:

https://github.com/johandahlberg/BestPracticeSlides

Example - Powerset

Powerset = All subsets of a set (incl set)

```
(1 2 3) ==>
((1 2 3) (1 2) (1 3) (1) (2 3) (2) (3) NIL)
```

JavaScript

```
function powerset(ary) {
   var ps = [[]];
   for (var i=0; i < ary.length; i++) {
      for (var j = 0, len = ps.length; j < len; j++) {
         ps.push(ps[j].concat(ary[i]));
      }
   }
   return ps;
}

var res = powerset([1,2,3,4]);

load('json2.js');
print(JSON.stringify(res));
```

Clojure

```
(use '[clojure.contrib.combinatorics :only [subsets] ])
(def S #{1 2 3 4})
(subsets S)
```

Example - Factorial

C

```
int factorial(int n) {
   int result = 1;
   for (int i = 1; i <= n; ++i)
       result *= i;
   return result;
}
```

Scala

Functional

```
def factorial(n: Int) = if(n == 0) 1 else n * factorial(n-1)
```

Imperative

```
def factorial(n: Int)={
 var res = 1
 for(i <- 1 to n)
   res *=i
 res
}
```