CSCI 1370

APRIL 26, 2017

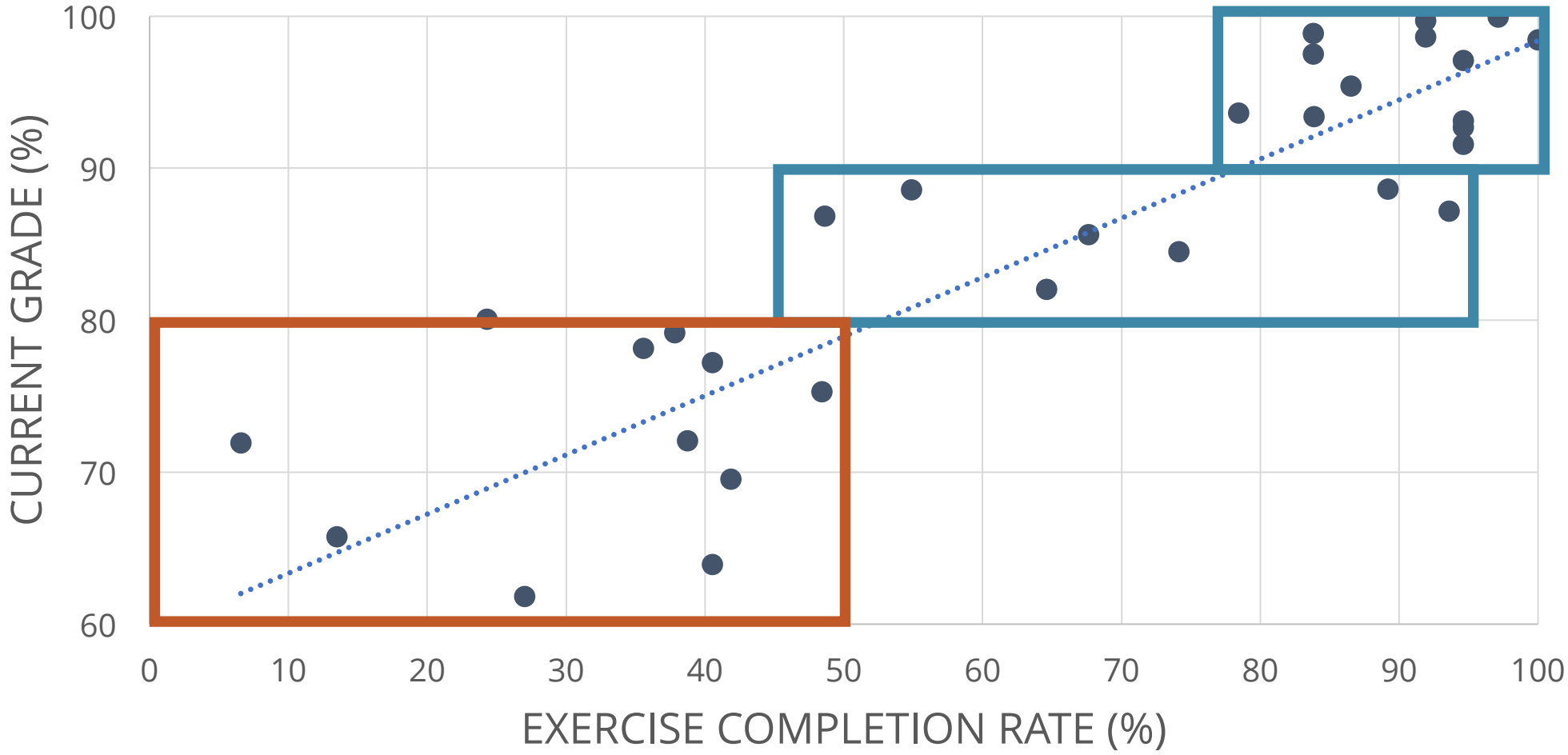
ADMINISTRATIVIA

Quarter Exam #3: scores ranged from 0.70 points to 10.05 points, with a median score of 7.07.

Note: a total bonus of 1.00 points (+.5 curve, +.5 group reward) was added. In addition, individually, a total of 3 points of extra credit was possible.

Note: five of the seven exam questions (plus two of the three extra credit questions) came from the set of exercises on repl.it.

CORRELATION BETWEEN COURSE GRADE (post-Exam 3) AND EXERCISE COMPLETION RATE



GRADES

Quizzes: scores have been updated to “participation points” (points for attendance).

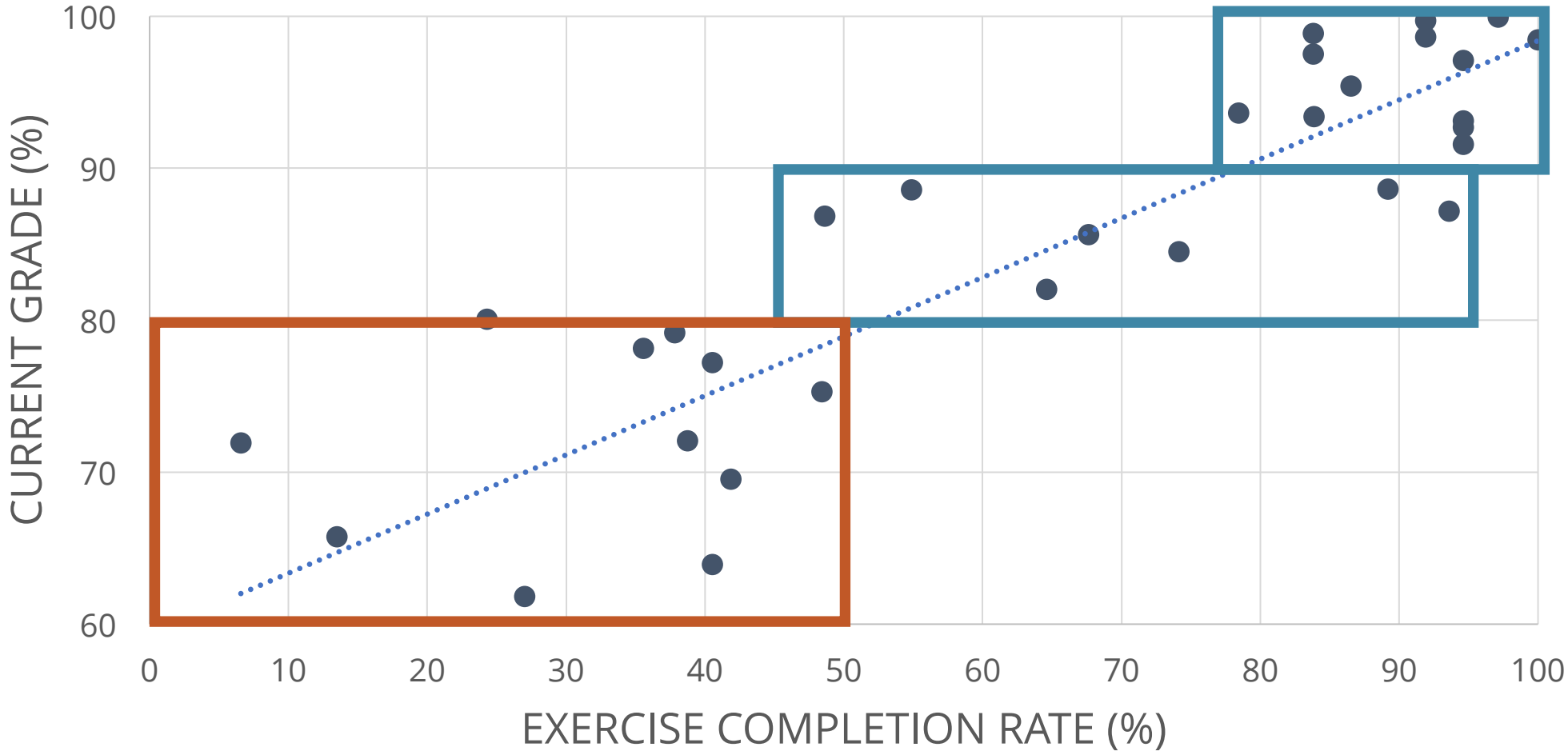
POINTS

Directly within student control: attendance (25 points), group meetings (7), peer evaluations (4), and repl.it exercises (14)...

Extra credit: exam bonuses (up to 6 points total), exam challenge questions (up to 8 points), extra lecture (up to 1 point)...

Indirectly within student control: quarter exams (30 points), final exam (20)...

CORRELATION BETWEEN COURSE GRADE (post-Exam 3) AND EXERCISE COMPLETION RATE



Blackboard shows current “max” grade possible in the course.

Expected grades can be computed as follows:

```
current grade – [20 – [[[average quarter exam score] / 10] * 20]]
```

CLASSES

The types `bool`, `char`, `double`, `float`, and `int` are all **primitive** data types.

The type , `string`, is different...specifically, `string` is a **class**.
(This is how we are able to store a sequence of characters.)

Classes allow for the grouping of relevant data and methods.

DECLARING CLASSES

Example:

```
class Student
{
  public:
      string name;
      string id;
};
```

Example:

```
class Student                    // keyword for
{                                // declaring a class
  public:
      string name;
      string id;
};
```

Example:

```
class Student                    // name of the class
{
  public:
      string name;
      string id;
};
```

Example:

```
class Student
{
  public:
      string name;
      string id;
};
```

```
// body of the class
// declaration
```

Example:

```
class Student
{
  public:
      string name;
      string id;                    // relevant data
};
```

Example:

```
class Student
{
  public:                          // “privacy” (or lack
        string name;               // thereof) of the
        string id;                 // subsequent items
};
```

If some information is sensitive, we can make it **private** (i.e., hidden from the rest of the program).

Example:

```
class Student
{
  public:                          // “privacy” (or lack
      string name;                 // thereof) of the
      string id;                   // subsequent items
};
```

Example:

```
class Student
{
  public:
      string name;

  private:
      string id;
};
```

Example:

```
class Student
{
  public:
      string name;

  private:                    // the variable, id, is
      string id;              // now private (visible
};                            // only within the
                              // student class)
```

In making a piece of data private, however, we now need ways (i.e., **methods**) of modifying it.

Example:

```
class Student
{
  public:
      string name;

  private:                    // the variable, id, is
      string id;              // now private (visible
};                            // only within the
                              // student class)
```

Example:

```
class Student
{
  public:
      string name;
      void setID(string s);
      string getID();

  private:
      string id;
};
```

Example:

```
class Student
{
  public:
      string name;
      void setID(string s);
      string getID();

  private:
      string id;
};
```

// class **methods**

DEFINING CLASS METHODS

```
class Student
{
   public:
         string name;
         void setID(string s);
         string getID();

   private:
         string id;
};

void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

```
class Student
{
   public:
          string name;
          void setID(string s);
          string getID();

   private:
          string id;
};
```

CLASS DECLARATION

```
void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

```
class Student
{
   public:
         string name;
         void setID(string s);
         string getID();

   private:
         string id;
};
```

```
void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

CLASS METHOD DEFINITIONS

```
class Student
{
   public:
         string name;
         void setID(string s);          // function prototype
         string getID();

   private:
         string id;
};

void Student::setID(string s)           // function header
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

```
class Student
{
   public:
         string name;
         void setID(string s);
         string getID();

   private:
         string id;
};

void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

// function definition

```
class Student
{
   public:
         string name;
         void setID(string s);
         string getID();                    // function prototype

   private:
         string id;
};

void Student::setID(string s)
{
   id = s;
}

string Student::getID()                     // function header
{
   return id;
}
```

```
class Student
{
   public:
         string name;
         void setID(string s);
         string getID();

   private:
         string id;
};

void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

// function definition

USING CLASSES

```cpp
// main.cpp

#include <iostream>
using namespace std;

#include “Student.h”

int main()
{
   Student s;

   s.name = “Berta”;
   s.setID(“123456789”);

   cout << s.getID();
}
```

```cpp
// Student.h

class Student
{
   public:
         string name;
         void setID(string s);
         string getID();

   private:
         string id;
};

void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

```cpp
// main.cpp

#include <iostream>
using namespace std;

#include “Student.h”

int main()
{
   Student s;

   s.name = “Berta”;
   s.setID(“123456789”);

   cout << s.getID();
}
```

```cpp
// Student.h

class Student
{
   public:
         string name;
         void setID(string s);
         string getID();

   private:
         string id;
};

void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

To modify data associated with a class instance (e.g., Student s), we use the syntax:

classInstance.classVariable

```cpp
// main.cpp

#include <iostream>
using namespace std;

#include “Student.h”

int main()
{
   Student s;

   s.name = “Berta”;
   s.setID(“123456789”);

   cout << s.getID();
}
```

```cpp
// Student.h

class Student
{
   public:
         string name;
         void setID(string s);
         string getID();

   private:
         string id;
};

void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

```cpp
// main.cpp

#include <iostream>
using namespace std;

#include “Student.h”

int main()
{
   Student s;

   s.name = “Berta”;
   s.setID(“123456789”);

   cout << s.getID();
}
```

```cpp
// Student.h

class Student
{
   public:
         string name;
         void setID(string s);
         string getID();

   private:
         string id;
};

void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

To modify data associated with a class instance (e.g., Student s), we use the syntax:

classInstance.classMethod(function parameters)

```cpp
// main.cpp

#include <iostream>
using namespace std;

#include “Student.h”

int main()
{
   Student s;

   s.name = “Berta”;
   s.setID(“123456789”);

   cout << s.getID();
}
```

```cpp
// Student.h

class Student
{
   public:
         string name;
         void setID(string s);
         string getID();

   private:
         string id;
};

void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

```cpp
// main.cpp

#include <iostream>
using namespace std;

#include “Student.h”

int main()
{
   Student s;

   s.name = “Berta”;
   s.setID(123456789);

   cout << s.getID();
}
```

```cpp
// Student.h

class Student
{
   public:
         string name;
         void setID(string s);
         string getID();

   private:
         double id;
};

void Student::setID(string s)
{
   id = s;
}

string Student::getID()
{
   return id;
}
```

Exercise: declare an *Employee class*.

Note: employees should have a name and an employee ID associated with them. Also note, IDs are sensitive info, so the data should be *private*. (Which means too, we need methods for getting and setting the value of the ID variable.)

Exercise: create an instance of the `Employee` class.

Specifically, in `main`: (a) declare a variable of type `Employee`, (b) assign the employee a name (e.g., “Professor Strait”) and ID (e.g., 111111111), and (c) print the ID value to the console.