# Programming I – algorithms and loops

How computers think
How to program them to think for you

# Computers are dumb but fast

- Computers are literal
  - They will do exactly what you tell them to do
  - They will not do what you don't tell them to do
- Computers are fast
  - They execute millions of instructions millions of times per second
- The trick in programming is telling the computer exactly what it needs to do to accomplish a task

# Algorithms

- Computers execute instructions one at a time
- Algorithms are step by step procedures for calculations
- They describe a series of steps for accomplishing a task
- To get a computer to do a task, we need to:
  - Identify the task to be completed
  - Figure out how to do the task using operations the computer knows how to do
  - Write instructions to the computer to do the steps

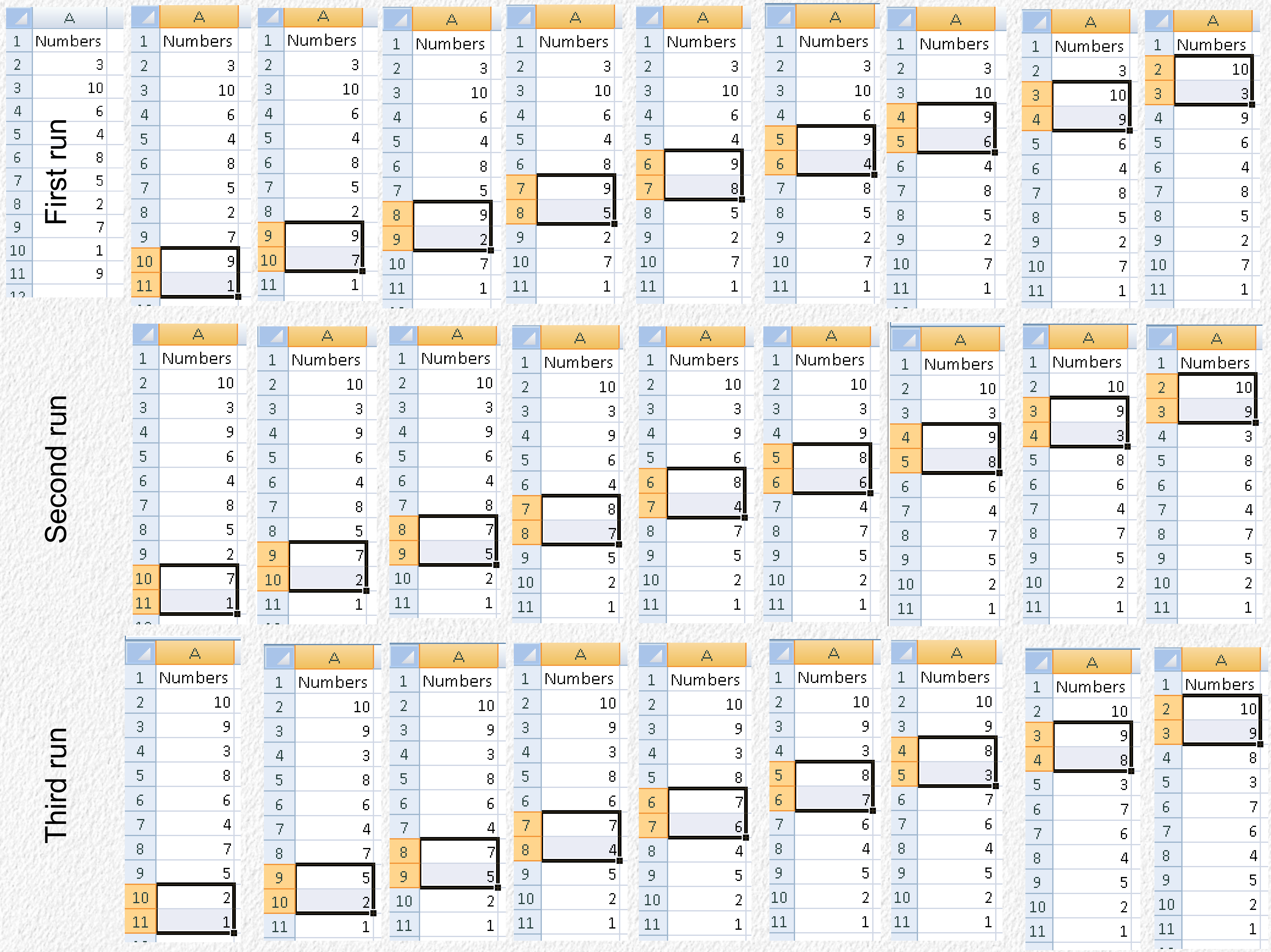
# Example: sorting numbers

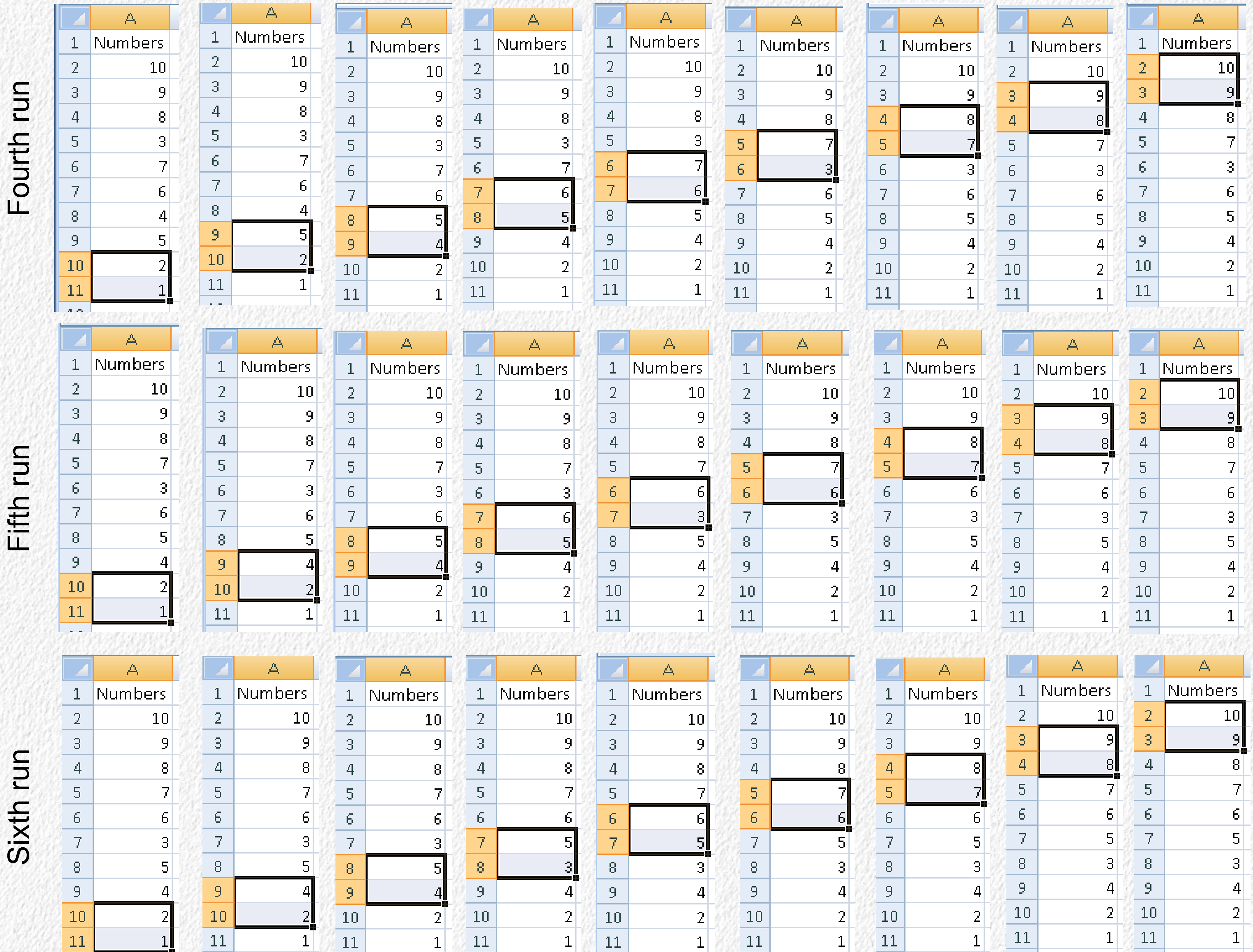
- How to sort numbers from 1 to 10 in descending order?
- There are many ways to do this
  - All will accomplish the task
  - Some will take longer than others

# “Bubble” sort

| | A |
|---|---|
| 1 | Numbers |
| 2 | 3 |
| 3 | 10 |
| 4 | 6 |
| 5 | 4 |
| 6 | 8 |
| 7 | 5 |
| 8 | 2 |
| 9 | 7 |
| 10 | 1 |
| 11 | 9 |

- Start with unsorted numbers
- Compare the first and second numbers – if they are out of order swap them
- Continue to second and third number, third and fourth, fourth and fifth, etc. until all comparisons have been made
- Repeat until no more swaps are needed


First run
Second run
Third run


Fourth run
Fifth run
Sixth run

## Seventh run

| | A | A | A | A | A | A | A | A | A |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Numbers | Numbers | Numbers | Numbers | Numbers | Numbers | Numbers | Numbers | Numbers |
| 2 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 3 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| 4 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 5 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 7 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 8 | 3 | 3 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 9 | 4 | 4 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 11 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

## Eighth run

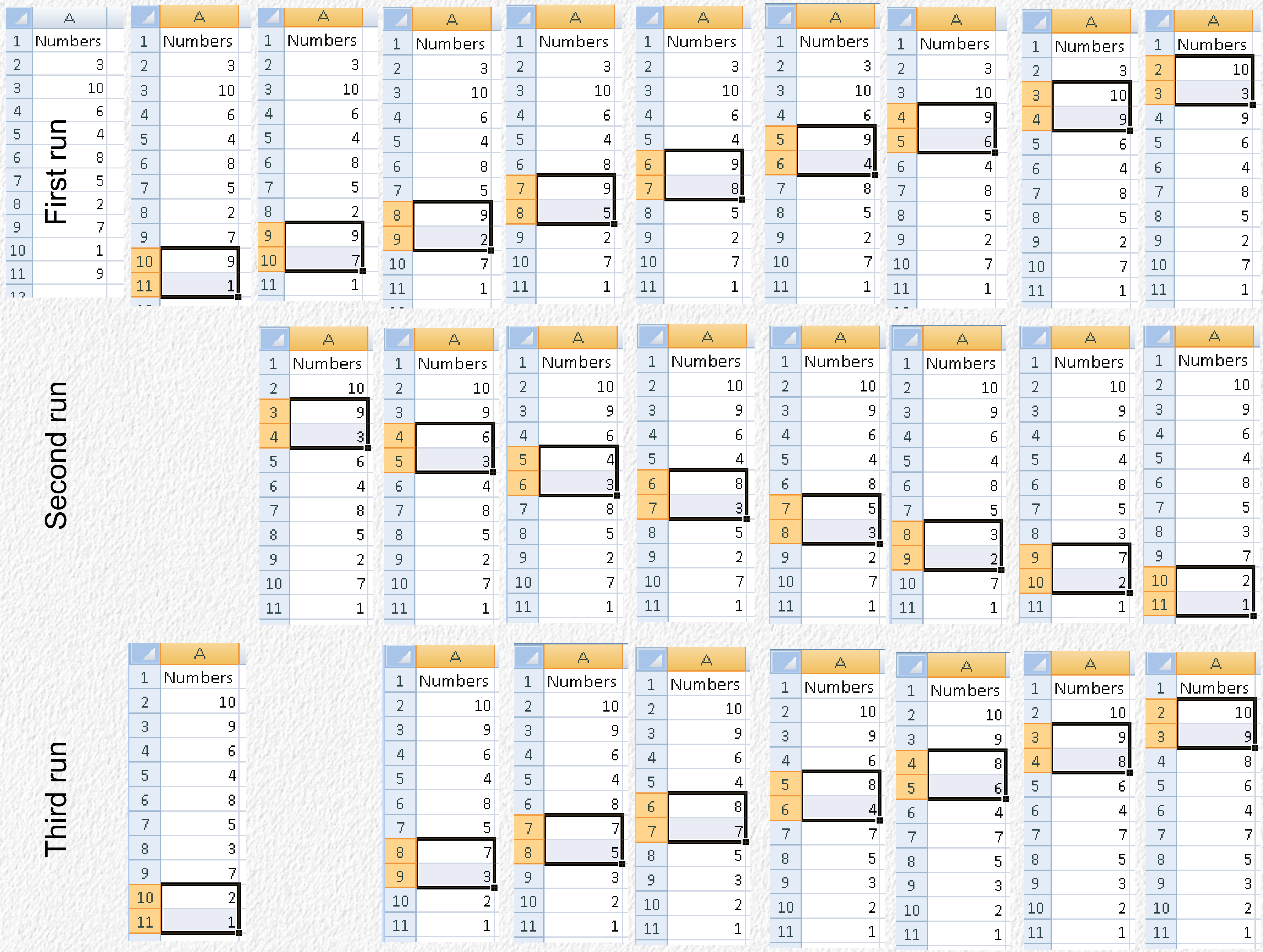
| | A | A | A | A | A | A | A | A | A |
|---|---|---|---|---|---|---|---|---|---|
| 1 | Numbers | Numbers | Numbers | Numbers | Numbers | Numbers | Numbers | Numbers | Numbers |
| 2 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 | 10 |
| 3 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 | 9 |
| 4 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 | 8 |
| 5 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 | 7 |
| 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 | 6 |
| 7 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 | 5 |
| 8 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 | 4 |
| 9 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 10 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 11 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

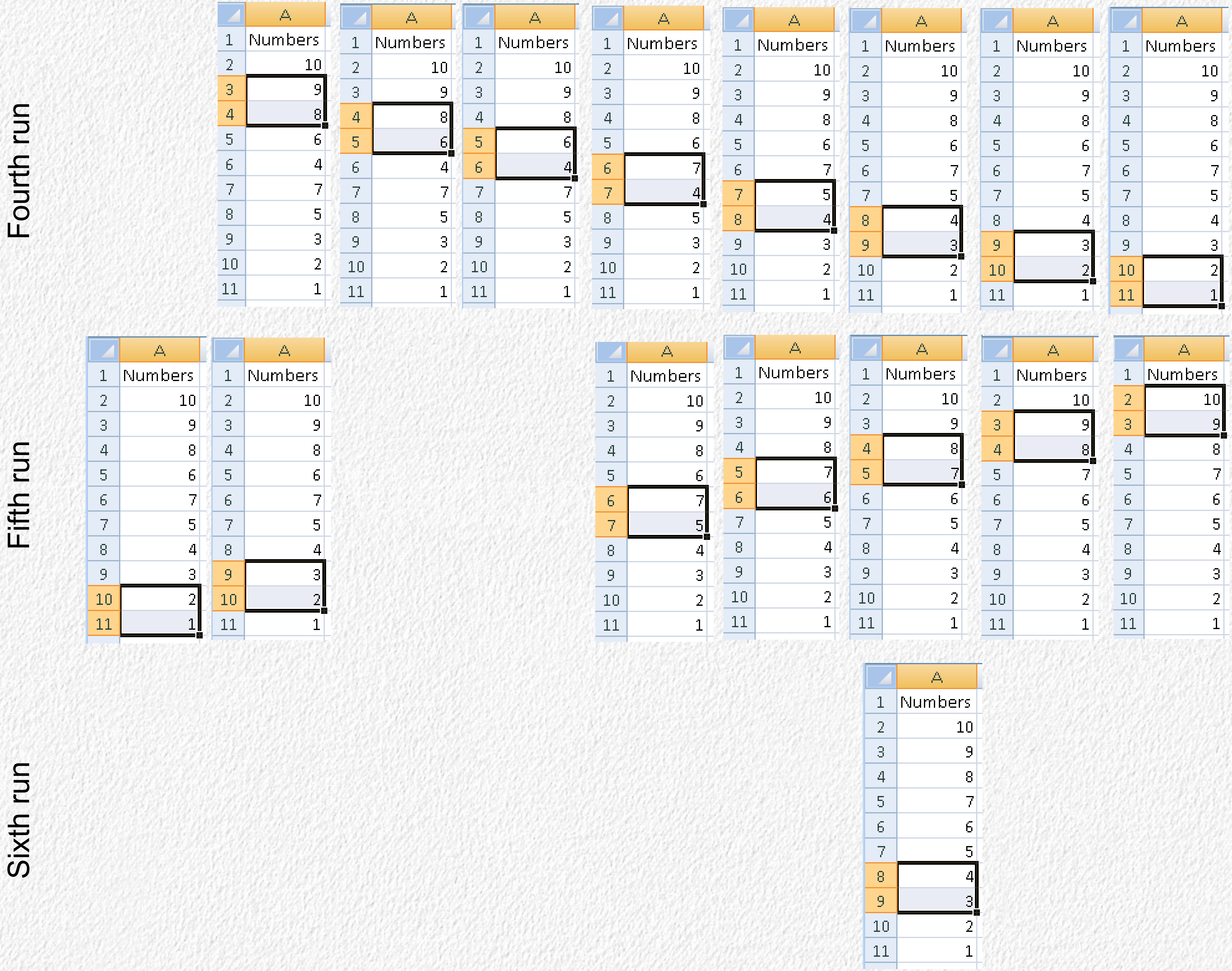
Finished!

Nothing changed in the last run, but needed to confirm that the list is in order

# Better bubble sort

- A weakness of the algorithm: numbers can move up rapidly, but down slowly
- An improved algorithm: alternate running up and down between runs


First run
Second run
Third run


Fourth run
Fifth run
Sixth run

# Programming a computer

- A program is a series of instructions executed by the computer
- They are written in a programming language
- They are executed in order, first to last

# Programming languages

- Many out there, some more “English like” in their syntax than others
- Written as “code” = a series of instructions, with a syntax specific to the language
- Some then execute the code from within an “interpreter” = another program that translates the code into a binary form the computer understands
- Some “compile” the program = convert it into a binary code the computer understands, which can then be run without an interpreter

# The language we will use

- The language used to program Excel is “Microsoft Visual Basic for Applications” (VBA)
  - Interpreted language
  - Only runs from within an MS Office application, but can take advantage of the capabilities of Excel
- Visual Basic is fairly simple to use, fairly English-like in its syntax
- Programs that run in Excel are called VBA “macros”

# Macros in Excel

- Three major uses
  - Automating a complex task
  - Automating a repetitive task
  - Implementing functions/algorithms not already available as functions in Excel
- Simplifies programming
  - Take advantage of Excel for storing data, file input/output, summary, graphing
- Constrained by the way Excel works
  - Need to learn to move around the worksheet, select cells, from within the program

# Automating a complex task

- Some operations take multiple steps to complete
- May be faster to:
  - Record yourself doing the task once with the macro recorder
  - Assign the macro to a key
  - Hit the key to run the macro and perform the task
- Example: setting the format on some cells...

# Automating a repetitive task

- If you need to do an operation on each cell in your spreadsheet one at a time, it may be faster to record the operation once, then write a “loop” to repeat it
- Repetitive task example – another sort algorithm

# Random re-ordering

- Let's try another sort algorithm
- The algorithm matters...how would random re-orderings work?
  - Start with the numbers
  - Generate random numbers
  - Sort by the random numbers
  - Check if the sort order is correct
- Let's try it once in Excel...

# Repeating tasks in a computer – using loops

- Loops are ways of telling the computer to repeat an operation until a condition is met
- The condition can be several different things:
  - A fixed number of repeats
  - A run through a list of arguments
  - A criteria that needs to be satisfied
- Begin and end with key words
- Details of the syntax of loops depends on the programming language

# Do loops

- Do loops can have two different forms:
  - Do while
    …
    loop
  - Do
    …
    loop until
- The criterion can be tested before or after the loop instructions within the loop are executed
- Once the criterion is met the program leaves the loop and continues on to the next instruction

# The code that Excel recorded for random sorter

```
Sub RandSort()
'
' RandSort Macro
' Show how slow a sorting algorithm that selects random sort orders would be.
'
' Keyboard Shortcut: Ctrl+Shift+R
'
    Range("A1:B11").Select
    ActiveWorkbook.Worksheets("RandomSorter").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("RandomSorter").Sort.SortFields.Add Key:=Range("B2:B11") _
        , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("RandomSorter").Sort
        .SetRange Range("A1:B11")
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
End Sub
```

# The header

```
'
' RandSort Macro
' Show how slow a sorting algorithm that selects random sort orders would be.
'
' Keyboard Shortcut: Ctrl+Shift+R
'
```

The apostrophes are “comment” characters. Anything after them is ignored by the interpreter.

Used to make notes about what the program does.

Comments are a Good Thing.

# Subroutines are marked by Sub, End Sub

```
Sub RandSort()

Instructions to execute...

End Sub
```

At least one must be present for the macro to run.

# Sorting by the random numbers

```
Range("A1:B11").Select

   ActiveWorkbook.Worksheets("RandomSorter").Sort.SortFields.Clear

   ActiveWorkbook.Worksheets("RandomSorter").Sort.SortFields.Add Key:=Range("B2:B11") _
      , SortOn:=xlSortOnValues, Order:=xlAscending,

DataOption:=xlSortNormal
   With ActiveWorkbook.Worksheets("RandomSorter").Sort
      .SetRange Range("A1:B11")
      .Header = xlYes
      .MatchCase = False
      .Orientation = xlTopToBottom
      .SortMethod = xlPinYin
      .Apply
   End With
```

*Select the range of data to sort (not needed)*

*Clear out any old sort keys*

*Identify the sort key to use, and the order (ascending)*

*Execute the sort*

# Now, make it run repeatedly

- The rand() function selects a new set of random numbers each time the sheet recalculates
- Sorting recalculates the sheet
- As soon as the numbers are sorted, there are new random numbers for sorting again
- All that's needed is to tell the macro to repeat the operation until the numbers are in order

# Do this repeatedly with a Do while...loop

```
Sub RandSort()
'
' RandSort Macro
' Show how slow a sorting algorithm that selects random sort orders would be.
'
' Keyboard Shortcut: Ctrl+Shift+R
'
Do While Range(“B13”) = False
    Range("A1:B11").Select
    ActiveWorkbook.Worksheets("RandomSorter").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("RandomSorter").Sort.SortFields.Add Key:=Range("B2:B11") _
        , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("RandomSorter").Sort
        ....
    End With
    Range(“B15”).Select
    ActiveCell.Value = ActiveCell.Value + 1
Loop

End Sub
```

***The loop***

***Count iterations***

# For...next loops

- Useful for executing an operation on a defined list of inputs, or a fixed number of times
- Syntax is:

```
For i in 1 to 10
   Things for the loop to do
Next i
```

- We'll use these a lot for randomization testing and bootstrapping

# Infinite loops

- Avoid these
- If you use a loop in which the ending condition cannot ever be met, it will run forever
- If this happens, VB allows you to interrupt a running macro
- Childishly easy to create!

# An infinite Do...loop

- The following is an infinite Do loop

  Range(“A2”).Value = False
  Do while Range(“A2”) = False

  Loop

- A2 is never changed, so it can never become True
- This will execute forever, until you stop it or the computer dies
- Stop a program with the Escape key (Esc)

# Randomization testing

- A “nonparametric” approach to analyzing data
- Generally used when the usual parametric approaches (t-tests, ANOVA, regression, etc.) aren't appropriate because of violated assumptions
- The sampling distribution is derived by randomly shuffling the data

# Example: Mantel tests

- Mantel tests are tests of association between two square matrices
- Often these are “distance matrices”
  - Geographic distance between sampled populations, genetic distance between sampled populations
- A measure of association between the matrices is calculated, then the elements of the matrix are randomly shuffled
- The association is re-calculated with each random shuffle
- The observed association is compared with the randomly generated differences to obtain a p-value

# Association between geographic distance and genetic distance

- Organisms tend to find mates in their vicinities
- This leads to “isolation by distance”
- Gene pools tend to become more different with increasing distance
- Is this true for humans?
- Let's look at the association between gene frequencies and location from the DNA fingerprint data

# The analysis

- Data from 7 states
- Calculate a genetic distance among all possible pairs of states
- Treat the location of the capitol city as the location, calculate distances among them
- Test for association using a Mantel test

# Euclidean distance

- As you no doubt recall, the distance between two points with coordinates $(x_1,y_1)$ and $(x_2, y_2)$ is:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

- If we have more than two coordinates we just continue to add squared differences:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2 + (z_1 - z_2)^{2.}..}$$

# Distance between capitols

| | California | Alabama | Florida | Virginia | New York | Michigan | Minnesota |
|---|---|---|---|---|---|---|---|
| California | | 2432.8299 | 2569.85796 | 3046.0153 | 3298.369 | 2553.161 | 1964.8485 |
| Alabama | | | 139.48841 | 613.4105 | 871.5 | 132.394 | 467.9823 |
| Florida | | | | 478.3441 | 740.6782 | 84.82056 | 605.4398 |
| Virginia | | | | | 268.7998 | 494.7943 | 1081.2876 |
| New York | | | | | | 746.3383 | 1336.8301 |
| Michigan | | | | | | | 590.5349 |
| Minnesota | | | | | | | |

*Done in another program – earth is curved, longitude lines are not parallel, need software that knows this*

# Distances between sets of gene frequencies

***California***

| Locus | Allele 1 | Allele 2 |
|---|---|---|
| D3S1358 | 0.2800 | 0.2167 |
| VWA | 0.2333 | 0.2800 |
| FGA | 0.1500 | 0.1767 |
| D8S1179 | 0.3733 | 0.3733 |
| D21S11 | 0.1967 | 0.2321 |
| D18S51 | 0.1467 | 0.1600 |
| D5S818 | 0.3400 | 0.3600 |
| D13S317 | 0.3133 | 0.2767 |
| D7S820 | 0.2433 | 0.2233 |
| THO1 | 0.2200 | 0.3233 |
| TPOX | 0.5267 | 0.5267 |
| CSF1PO | 0.3005 | 0.3251 |

***Alabama***

| Locus | Allele 1 | Allele 2 |
|---|---|---|
| D3S1358 | 0.2300 | 0.2567 |
| VWA | 0.2133 | 0.2800 |
| FGA | 0.1367 | 0.1900 |
| D8S1179 | 0.3133 | 0.3133 |
| D21S11 | 0.1867 | 0.2733 |
| D18S51 | 0.1567 | 0.1100 |
| D5S818 | 0.4167 | 0.3667 |
| D13S317 | 0.3200 | 0.2667 |
| D7S820 | 0.2967 | 0.1500 |
| THO1 | 0.1967 | 0.3067 |
| TPOX | 0.5433 | 0.5433 |
| CSF1PO | 0.3033 | 0.3200 |

M10 {=SUM(($B4:$B15-C4:C15)^2+($B21:$B32-C21:C32)^2)}

Allele 1

| Locus | CA | AL | FL | VA | NY | MI | MN |
|---|---|---|---|---|---|---|---|
| D3S1358 | 0.2800 | 0.2300 | 0.2736 | 0.2437 | 0.2943 | 0.2844 | 0.2833 |
| VWA | 0.2333 | 0.2133 | 0.2093 | 0.2284 | 0.2270 | 0.2188 | 0.2300 |
| FGA | 0.1500 | 0.1367 | 0.1592 | 0.1472 | 0.1099 | 0.1656 | 0.1300 |
| D8S1179 | 0.3733 | 0.3133 | 0.3557 | 0.3112 | 0.3156 | 0.3469 | 0.3000 |
| D21S11 | 0.1967 | 0.1867 | 0.2289 | 0.2117 | 0.2179 | 0.2500 | 0.1867 |
| D18S51 | 0.1467 | 0.1567 | 0.1617 | 0.1556 | 0.1879 | 0.1625 | 0.1633 |
| D5S818 | 0.3400 | 0.4167 | 0.3740 | 0.3858 | 0.3794 | 0.3600 | 0.3633 |
| D13S317 | 0.3133 | 0.3200 | 0.3232 | 0.3173 | 0.3475 | 0.3576 | 0.2967 |
| D7S820 | 0.2433 | 0.2967 | 0.2622 | 0.3147 | 0.3156 | 0.2848 | 0.2700 |
| THO1 | 0.2200 | 0.1967 | 0.2378 | 0.2411 | 0.2113 | 0.2143 | 0.2467 |
| TPOX | 0.5267 | 0.5433 | 0.5488 | 0.5254 | 0.5458 | 0.5408 | 0.5600 |
| CSF1PO | 0.3005 | 0.3033 | 0.3679 | 0.2944 | 0.2993 | 0.3151 | 0.2933 |

Genetic distances

| | California | Alabama | Florida | Virginia | New York | Michigan | Minnesota |
|---|---|---|---|---|---|---|---|
| California | | 0.032152 | 0.01559 | 0.023229 | 0.030644 | 0.016594 | 0.02409 |
| Alabama | | | 0.022855 | 0.013359 | 0.023823 | 0.024248 | 0.015337 |
| Florida | | | | 0.023953 | 0.028312 | 0.012459 | 0.021753 |
| Virginia | | | | | 0.012674 | 0.01529 | 0.015625 |
| New York | | | | | | 0.015094 | 0.022419 |
| Michigan | | | | | | | 0.02146 |
| Minnesota | | | | | | | |

Allele 2

| Locus | CA | AL | FL | VA | NY | MI | MN |
|---|---|---|---|---|---|---|---|
| D3S1358 | 0.2167 | 0.2567 | 0.2338 | 0.2563 | 0.2563 | 0.2482 | 0.2375 |
| VWA | 0.2800 | 0.2800 | 0.2967 | 0.2792 | 0.2908 | 0.2844 | 0.2567 |
| FGA | 0.1767 | 0.1900 | 0.1642 | 0.1675 | 0.1738 | 0.1625 | 0.1667 |
| D8S1179 | 0.3733 | 0.3133 | 0.3557 | 0.3112 | 0.3156 | 0.3469 | 0.3000 |
| D21S11 | 0.2321 | 0.2733 | 0.2786 | 0.2143 | 0.2143 | 0.2156 | 0.2700 |
| D18S51 | 0.1600 | 0.1100 | 0.1318 | 0.1480 | 0.1170 | 0.1188 | 0.1167 |
| D5S818 | 0.3600 | 0.3667 | 0.3557 | 0.3350 | 0.2801 | 0.3433 | 0.3633 |
| D13S317 | 0.2767 | 0.2667 | 0.2541 | 0.2766 | 0.2766 | 0.2881 | 0.3067 |
| D7S820 | 0.2233 | 0.1500 | 0.2093 | 0.1777 | 0.1986 | 0.1987 | 0.1833 |
| THO1 | 0.3233 | 0.3067 | 0.2947 | 0.2944 | 0.3063 | 0.2789 | 0.2900 |
| TPOX | 0.5267 | 0.5433 | 0.5488 | 0.5254 | 0.5458 | 0.5408 | 0.5600 |
| CSF1PO | 0.3251 | 0.3200 | 0.3069 | 0.3477 | 0.3345 | 0.2979 | 0.3200 |

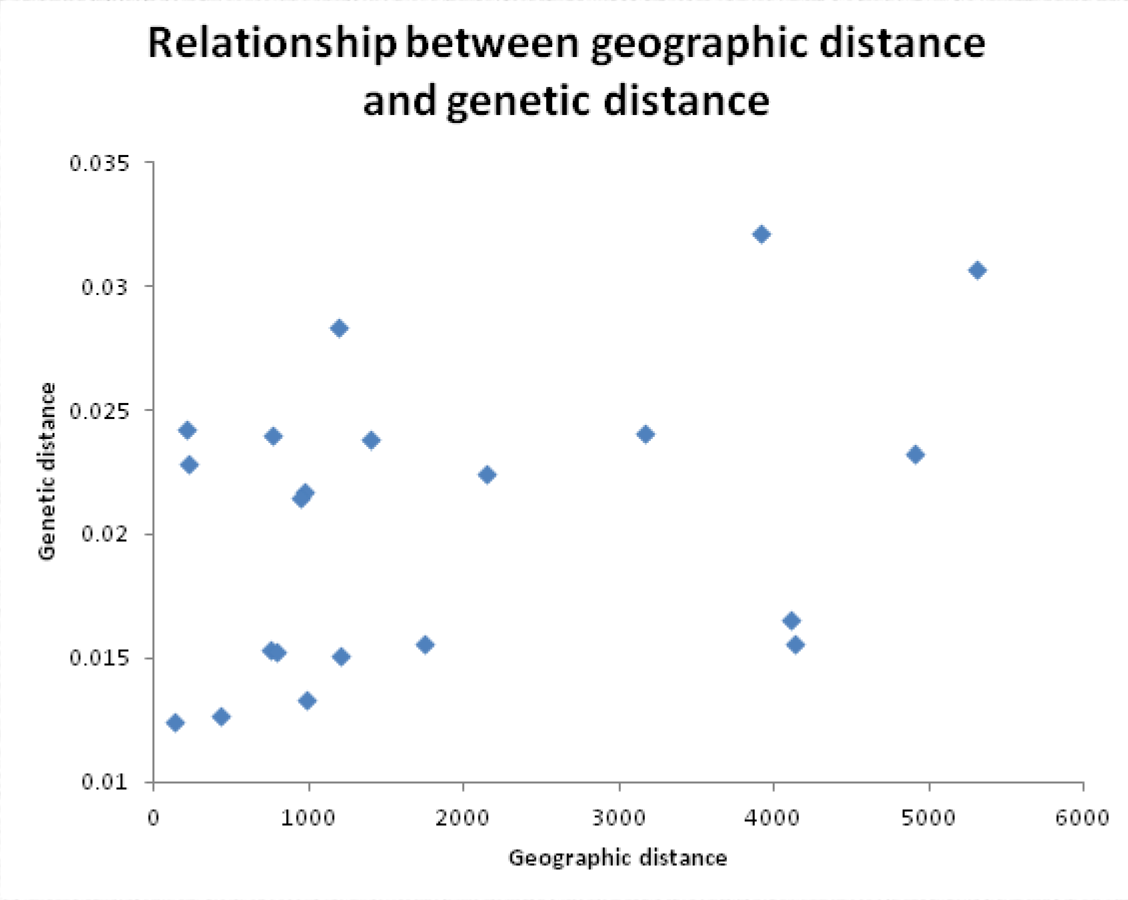
Genetic distances

| | California | Alabama | Florida | Virginia | New York | Michigan | Minnesota |
|---|---|---|---|---|---|---|---|
| California | | 0.032152 | 0.01559 | 0.023229 | 0.030644 | 0.016594 | 0.02409 |
| Alabama | | | 0.022855 | 0.013359 | 0.023823 | 0.024248 | 0.015337 |
| Florida | | | | 0.023953 | 0.028312 | 0.012459 | 0.021753 |
| Virginia | | | | | 0.012674 | 0.01529 | 0.015625 |
| New York | | | | | | 0.015094 | 0.022419 |
| Michigan | | | | | | | 0.02146 |
| Minnesota | | | | | | | |

Geographic distance (km)

| | California | Alabama | Florida | Virginia | New York | Michigan | Minnesota |
|---|---|---|---|---|---|---|---|
| California | | 3915.267 | 4135.792 | 4902.095 | 5308.219 | 4108.921 | 3162.122 |
| Alabama | | | 224.4852 | 987.1902 | 1402.546 | 213.0678 | 753.1457 |
| Florida | | | | 769.8215 | 1192.008 | 136.5057 | 974.3626 |
| Virginia | | | | | 432.5921 | 796.2956 | 1740.167 |
| New York | | | | | | 1201.117 | 2151.423 |
| Michigan | | | | | | | 950.3753 |
| Minnesota | | | | | | | |

# The relationship we'll test




***Correlation between these is 0.39***

Does the genetic distance depend on geographic distance?

# Why not just test the correlation?

- The measures aren't independent
- We have only 7 states, but we've generated 21 distances of each type
- Since parametric tests require independence, we can't use them
- But, a randomization test doesn't make this assumption, because any dependence will be accounted for when we randomly shuffle the data

# Unfold the data

Genetic distances

| | California | Alabama | Florida | Virginia | New York | Michigan | Minnesota |
|---|---|---|---|---|---|---|---|
| California | | 0.032152 | 0.01559 | 0.023229 | 0.030644 | 0.016594 | 0.02409 |
| Alabama | | | 0.022855 | 0.013359 | 0.023823 | 0.024248 | 0.015337 |
| Florida | | | | 0.023953 | 0.028312 | 0.012459 | 0.021753 |
| Virginia | | | | | 0.012674 | 0.01529 | 0.015625 |
| New York | | | | | | 0.015094 | 0.022419 |
| Michigan | | | | | | | 0.02146 |
| Minnesota | | | | | | | |

Geographic distance (km)

| | California | Alabama | Florida | Virginia | New York | Michigan | Minnesota |
|---|---|---|---|---|---|---|---|
| California | | 3915.267 | 4135.792 | 4902.095 | 5308.219 | 4108.921 | 3162.122 |
| Alabama | | | 224.4852 | 987.1902 | 1402.546 | 213.0678 | 753.1457 |
| Florida | | | | 769.8215 | 1192.008 | 136.5057 | 974.3626 |
| Virginia | | | | | 432.5921 | 796.2956 | 1740.167 |
| New York | | | | | | 1201.117 | 2151.423 |
| Michigan | | | | | | | 950.3753 |
| Minnesota | | | | | | | |

→

| | A | B | C |
|---|---|---|---|
| 1 | Comparison | Geograph | Genetic |
| 2 | California to Minnesota | 3162.122 | 0.02409 |
| 3 | Alabama to Minnesota | 753.1457 | 0.015337 |
| 4 | Florida to Minnesota | 974.3626 | 0.021753 |
| 5 | Virginia to Minnesota | 1740.167 | 0.015625 |
| 6 | New York to Minnesota | 2151.423 | 0.022419 |
| 7 | Michigan to Minnesota | 950.3753 | 0.02146 |
| 8 | California to Michigan | 4108.921 | 0.016594 |
| 9 | Alabama to Michigan | 213.0678 | 0.024248 |
| 10 | Florida to Michigan | 136.5057 | 0.012459 |
| 11 | Virginia to Michigan | 796.2956 | 0.01529 |
| 12 | New York to Michigan | 1201.117 | 0.015094 |
| 13 | California to New York | 5308.219 | 0.030644 |
| 14 | Alabama to New York | 1402.546 | 0.023823 |
| 15 | Florida to New York | 1192.008 | 0.028312 |
| 16 | Virginia to New York | 432.5921 | 0.012674 |
| 17 | California to Virginia | 4902.095 | 0.023229 |
| 18 | Alabama to Virginia | 987.1902 | 0.013359 |
| 19 | Florida to Virginia | 769.8215 | 0.023953 |
| 20 | California to Florida | 4135.792 | 0.01559 |
| 21 | Alabama to Florida | 224.4852 | 0.022855 |
| 22 | California to Alabama | 3915.267 | 0.032152 |

# The logic of the test

- Assume no relationship
  - The correlation between them is just random sampling
  - If so, the amount of correlation should be typical of randomly generated data
- If true, randomly shuffled genetic and geographic distances will give correlations as big as observed
- Conversely, if the amount of association we see is big compared to what we see when we randomly shuffle the data, we can conclude the association is real

# Set up the worksheet

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Comparison | Geograph | Genetic | Randomizer | | Sums of products |
| 2 | California to Minnesota | 3162.122 | 0.02408959 | 0.918175332 | | |
| 3 | Alabama to Minnesota | 753.1457 | 0.01533667 | 0.561645718 | | |
| 4 | Florida to Minnesota | 974.3626 | 0.02175256 | 0.083160037 | | |
| 5 | Virginia to Minnesota | 1740.167 | 0.01562474 | 0.213115326 | | |
| 6 | New York to Minnesota | 2151.423 | 0.0224185 | 0.973133831 | | |
| 7 | Michigan to Minnesota | 950.3753 | 0.02145971 | 0.048686382 | | |
| 8 | California to Michigan | 4108.921 | 0.01659358 | 0.608297796 | | |
| 9 | Alabama to Michigan | 213.0678 | 0.02424808 | 0.391236353 | | |
| 10 | Florida to Michigan | 136.5057 | 0.01245909 | 0.429935504 | | |
| 11 | Virginia to Michigan | 796.2956 | 0.01528989 | 0.068107105 | | |
| 12 | New York to Michigan | 1201.117 | 0.01509435 | 0.513797044 | | |
| 13 | California to New York | 5308.219 | 0.03064421 | 0.490451294 | | |
| 14 | Alabama to New York | 1402.546 | 0.02382347 | 0.879806774 | | |
| 15 | Florida to New York | 1192.008 | 0.02831232 | 0.159336885 | | |
| 16 | Virginia to New York | 432.5921 | 0.01267416 | 0.947817755 | | |
| 17 | California to Virginia | 4902.095 | 0.02322859 | 0.054737555 | | |
| 18 | Alabama to Virginia | 987.1902 | 0.01335929 | 0.273387711 | | |
| 19 | Florida to Virginia | 769.8215 | 0.02395328 | 0.426013632 | | |
| 20 | California to Florida | 4135.792 | 0.01559009 | 0.651453488 | | |
| 21 | Alabama to Florida | 224.4852 | 0.02285535 | 0.805971365 | | |
| 22 | California to Alabama | 3915.267 | 0.03215176 | 0.957294725 | | |
| 23 | | | | | | |
| 24 | Sum of products | | 886.3886249 | | | |
| 25 | (the Mantel test statistic) | | | | | |
| 26 | | | | | | |
| 27 | Observed sum of products | | 886.3886249 | | | |
| 28 | | | | | | |
| 29 | | | | | | |
| 30 | | | | | | |
| 31 | | | | | | |

Column for test statistic for random shuffles

=rand()

Use the macro recorder to:

Sort ***only the genetic distances*** by the Randomizer column

Copy the new test statistic and paste-special to column F

{=sum(product(b2:b22,c2:c22))}

A copy of the observed test statistic

# Macro as recorded

(General) | MantelTest

```
Sub MantelTest()
'
' MantelTest Macro
' Conduct a Mantel test on the geographic and genetic distances.
'
' Keyboard Shortcut: Ctrl+Shift+M
'
    Range("C1:D22").Select
    ActiveWorkbook.Worksheets("Sheet2").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Sheet2").Sort.SortFields.Add Key:=Range("D2:D22") _
        , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Sheet2").Sort
        .SetRange Range("C1:D22")
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
    Range("C24").Select
    Selection.Copy
    Range("F2").Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
End Sub
```

# Modify the macro to loop

- Two changes:
  - Add a “For...next” loop
  - Each time through the macro, need to store the measure of association
- Currently, copying/pasting measure of association to F2 – looping will make us replace this number each time through
- But, the counter (“i”) increases by 1 each iteration – if we paste to cell F(i+1), we will write to a new row each time

# Add a loop, record each result

```
Sub MantelTest()
'
' MantelTest Macro
' Conduct a Mantel test on the geographic and genetic distances.
'
' Keyboard Shortcut: Ctrl+Shift+M
'

For i = 1 To 1000
    Range("C1:D22").Select
    ActiveWorkbook.Worksheets("Sheet2").Sort.SortFields.Clear
    ActiveWorkbook.Worksheets("Sheet2").Sort.SortFields.Add Key:=Range("D2:D22") _
        , SortOn:=xlSortOnValues, Order:=xlAscending, DataOption:=xlSortNormal
    With ActiveWorkbook.Worksheets("Sheet2").Sort
        .SetRange Range("C1:D22")
        .Header = xlYes
        .MatchCase = False
        .Orientation = xlTopToBottom
        .SortMethod = xlPinYin
        .Apply
    End With
    Range("C24").Select
    Selection.Copy
    Range("F" & i + 1).Select
    Selection.PasteSpecial Paste:=xlPasteValues, Operation:=xlNone, SkipBlanks _
        :=False, Transpose:=False
Next i
End Sub
```

Select and copy the Mantel statistic

Select a location to record it

Paste-special the value

# Run the macro, sort the results

| F |
|---|
| Sums of products |
| 680.4899364 |
| 691.633857 |
| 694.7870141 |
| 695.480302 |
| 703.5632657 |
| 707.3579128 |
| 708.6603744 |
| 708.7029923 |
| 709.2008845 |
| 709.2259746 |
| 714.464126 |
| 716.5807235 |
| 719.5179922 |
| 903.9585579 |
| 908.6388034 |
| 909.1593751 |
| 909.5246373 |
| 909.8134634 |
| 910.7131166 |
| 912.239676 |
| 913.8250635 |
| 914.4788926 |
| 918.1006404 |
| 918.1550153 |
| 921.2731812 |
| 922.473072 |
| 927.1757834 |
| 932.7133386 |
| 937.300003 |

How many exceeded the observed?

We're only interested in whether there was a greater association than observed, so can just look at values bigger than observed (one-tailed test)

| | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 27 | Observed sum of products | | 886.388625 | | | 729.5221284 | |
| 28 | | | | | | 729.5840747 | |
| 958 | | | | | | 882.8188554 | |
| 959 | | | | | | 883.0791281 | |
| 960 | | | | | | 885.5101171 | |
| 961 | | | | | | 886.5968693 | |
| 962 | | | | | | 886.6395865 | |
| 963 | | | | | | 886.9566006 | |
| 964 | | | | | | 887.5672563 | |
| 965 | | | | | | 887.6931162 | |
| 966 | | | | | | 888.1998953 | |
| 967 | | | | | | 888.759341 | |
| 968 | | | | | | 889.0449391 | |
| 969 | | | | | | 891.3669924 | |
| 970 | | | | | | 891.7571737 | |
| 971 | | | | | | 892.8692841 | |
| 972 | | | | | | 893.0389275 | |
| 973 | | | | | | 894.7192509 | |
| 974 | | | | | | 895.6236515 | |
| 975 | | | | | | 895.8392815 | |
| 976 | | | | | | 896.0025476 | |
| 977 | | | | | | 896.2078633 | |
| 978 | | | | | | 896.4582357 | |
| 979 | | | | | | 897.1162263 | |
| 980 | | | | | | 899.7294216 | |
| 981 | | | | | | 900.0607245 | |
| 982 | | | | | | 900.2007867 | |
| 983 | | | | | | 901.6036995 | |
| 984 | | | | | | 903.2106488 | |
| 985 | | | | | | 903.6495732 | |
| 986 | | | | | | 903.9585579 | |
| 987 | | | | | | 908.6388034 | |
| 988 | | | | | | 909.1593751 | |
| 989 | | | | | | 909.5246373 | |
| 990 | | | | | | 909.8134634 | |
| 991 | | | | | | 910.7131166 | |
| 992 | | | | | | 912.239676 | |
| 993 | | | | | | 913.8250635 | |
| 994 | | | | | | 914.4788926 | |
| 995 | | | | | | 918.1006404 | |
| 996 | | | | | | 918.1550153 | |
| 997 | | | | | | 921.2731812 | |
| 998 | | | | | | 922.473072 | |
| 999 | | | | | | 927.1757834 | |
| 1000 | | | | | | 932.7133386 | |
| 1001 | | | | | | 937.300003 | |
| 1002 | | | | | | | |

# Calculate p

$$p = \frac{Number\ exceeding\ observed + 1}{Number\ of\ iterations + 1}$$

$$p = \frac{41 + 1}{1000 + 1} = 0.042$$

Reject the null – there is a (weak but) significant association between genetic distance and geographic distance