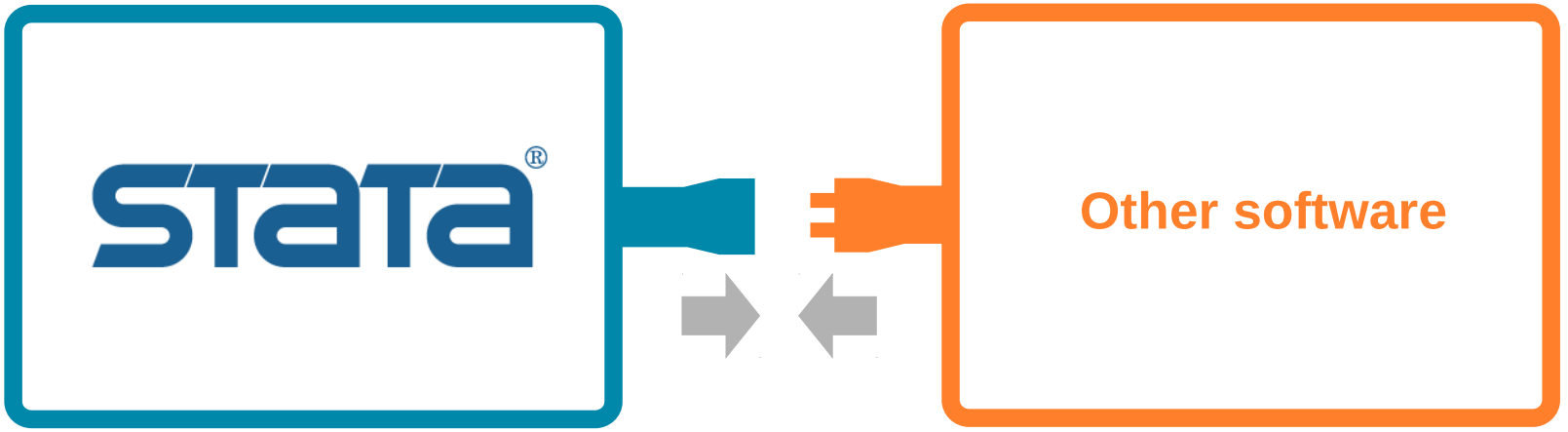
2017 German Stata Users Group meeting
Berlin, June 23

Connecting Stata to the rest of the world via SWire:

several applications including SWordy, an Office add-in to facilitate interaction between Microsoft Word and Stata

Giovanni Luca Lo Magno
lomagno.gl@virgilio.it
University of Palermo

SWire at a glance



- SWire stands for Stata-Wire
- remotely control Stata
- exchange data with other software

Requirement

Stata ≥ 13

Project web page

https://sourceforge.net/projects/swire/

Connect Stata to the world: several examples

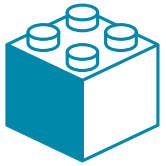
QGis

The R software

Internet browsers

Microsoft Office

What is SWire?



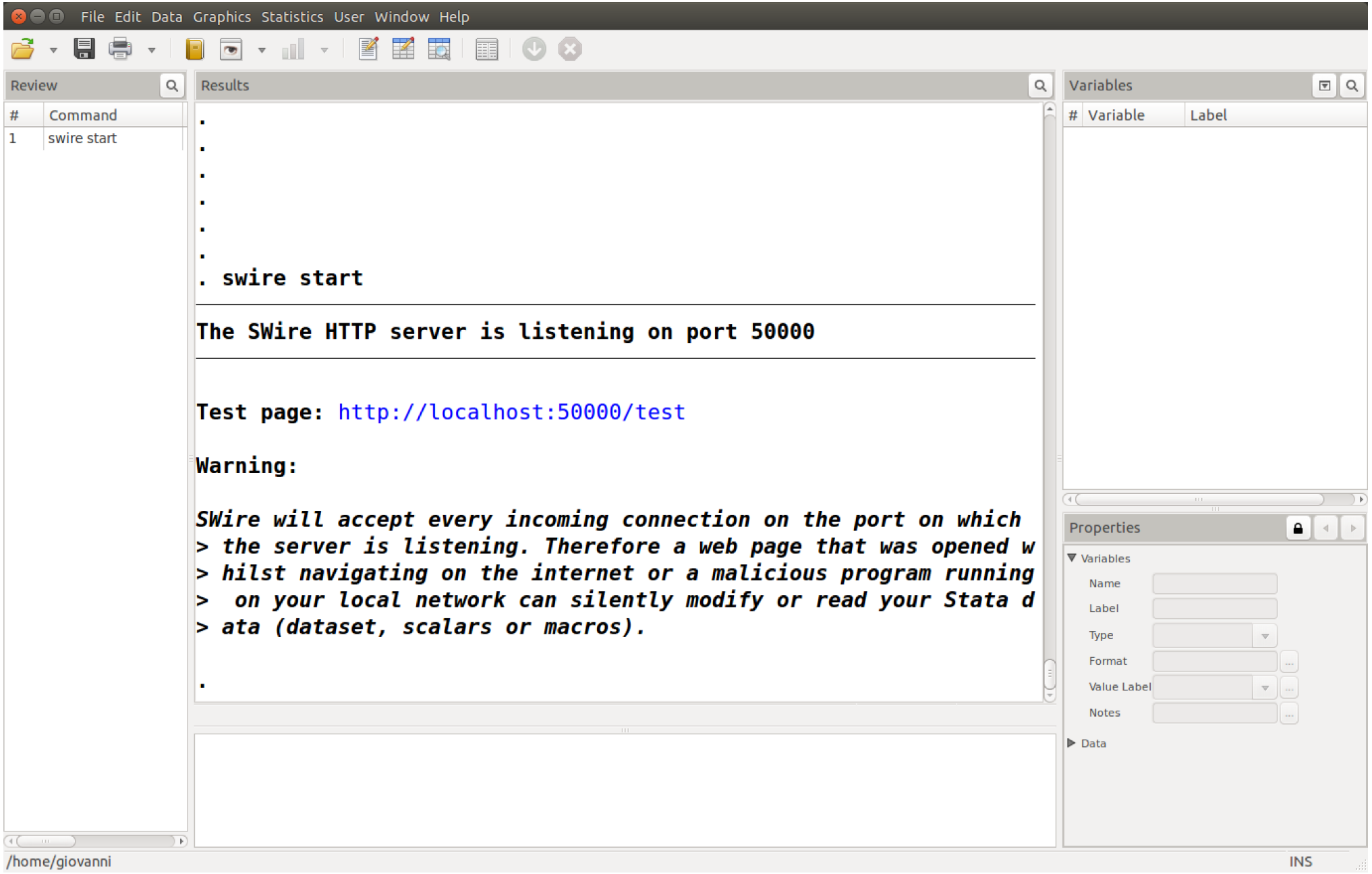
A Java plugin

A server running in Stata

A communication protocol

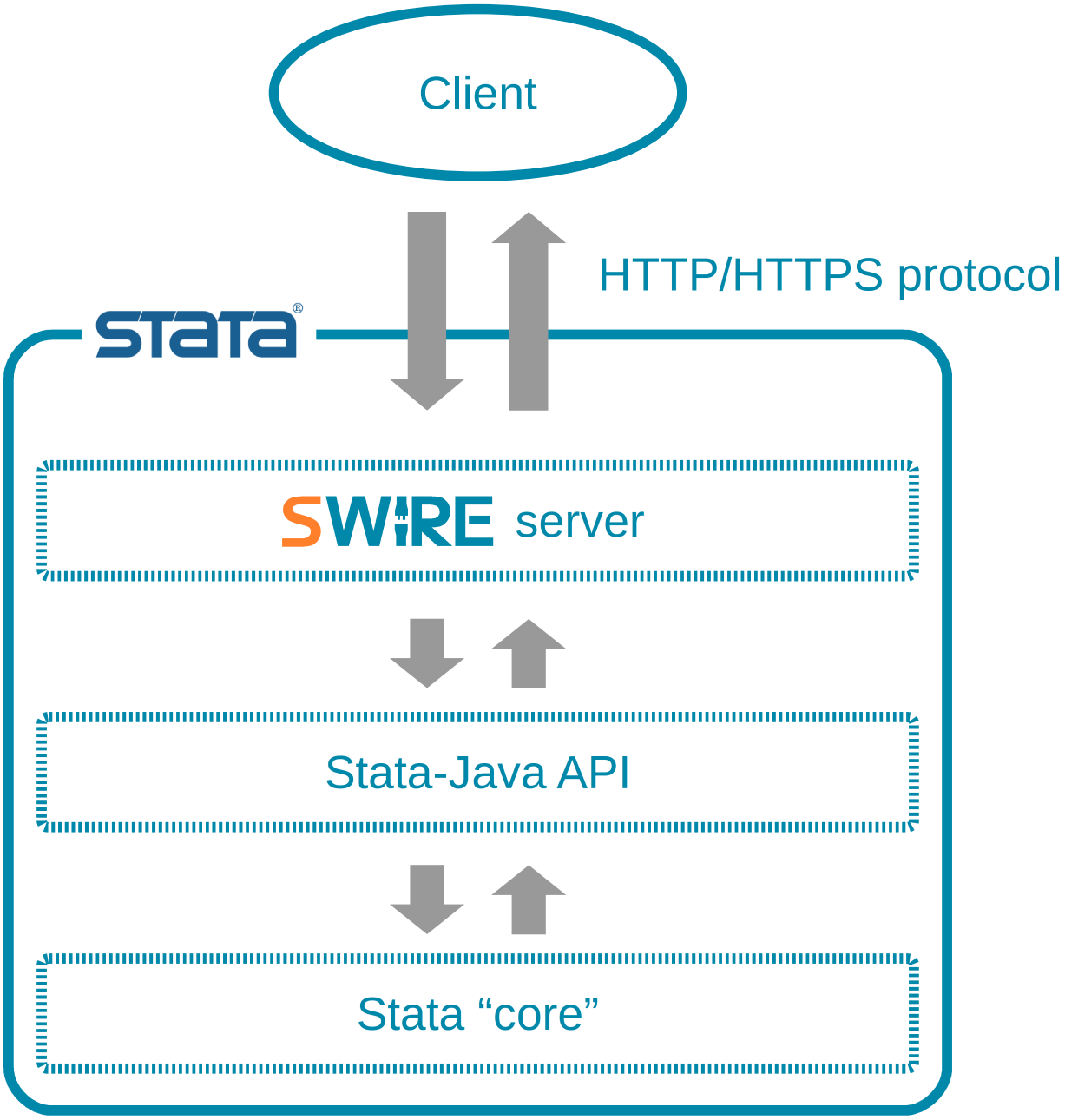
A developer tool

Starting the SWire server

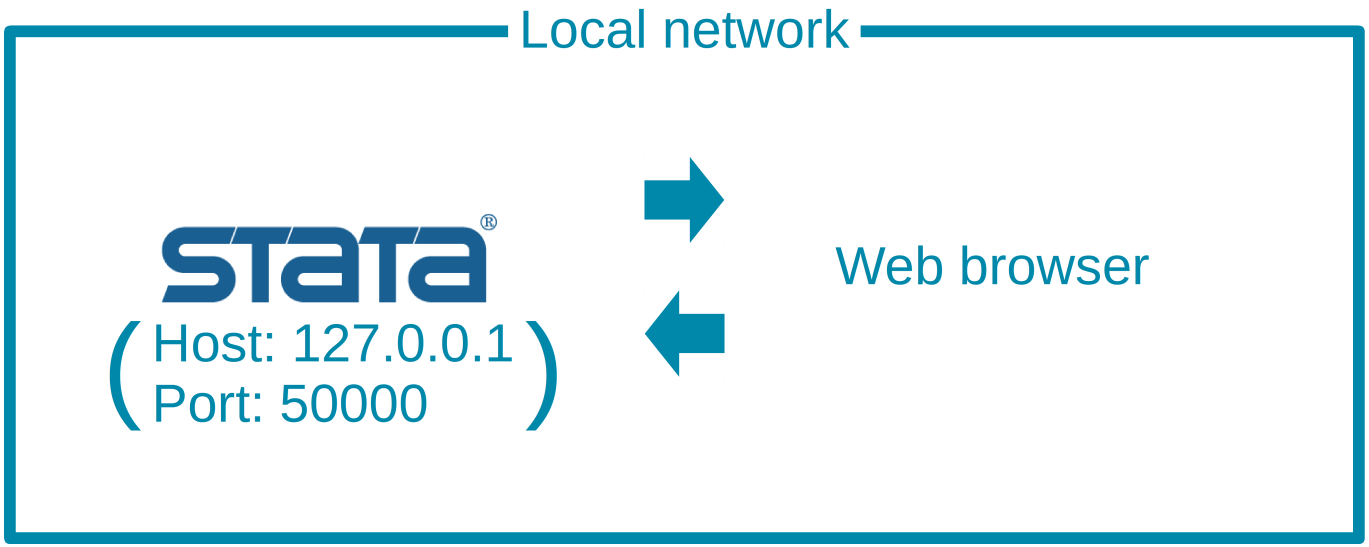


Note: the user can continue to use Stata while the server is listening

The SWire architecture



SWire web applications



- HTML/Javascript apps
- AJAX requests
- no browser refresh is required

SWire web apps from the Internet

No local installation

Data can be silently modified

SWire Web Apps Collection

A collection of demo SWire web apps:

- CoordPicker
- TV shows survey
- Automatic report

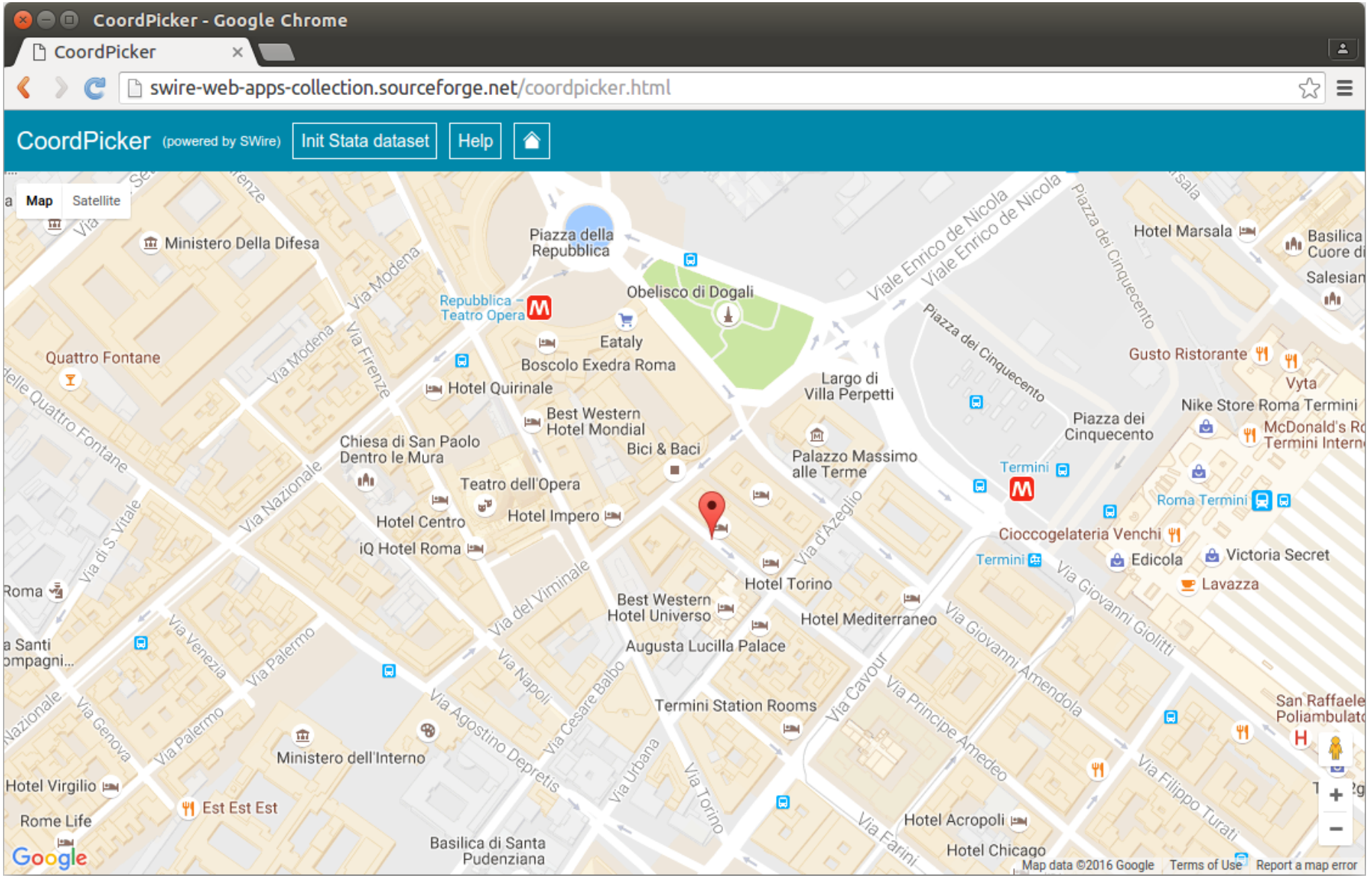
Project web page:

https://sourceforge.net/projects/swire-web-apps-collection/

Try it from the web:

http://swire-web-apps-collection.sourceforge.net/

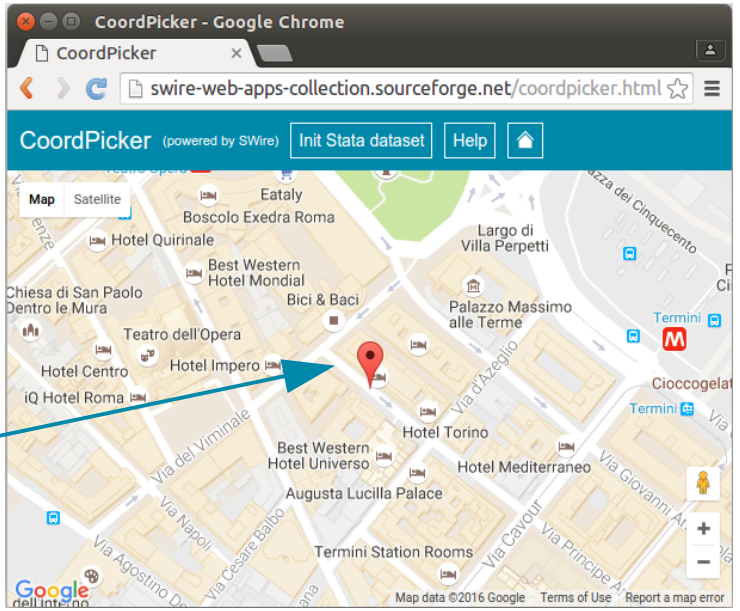
CoordPicker



Based on the Google Maps API

Pick latitude, longitude and address

CoordPicker



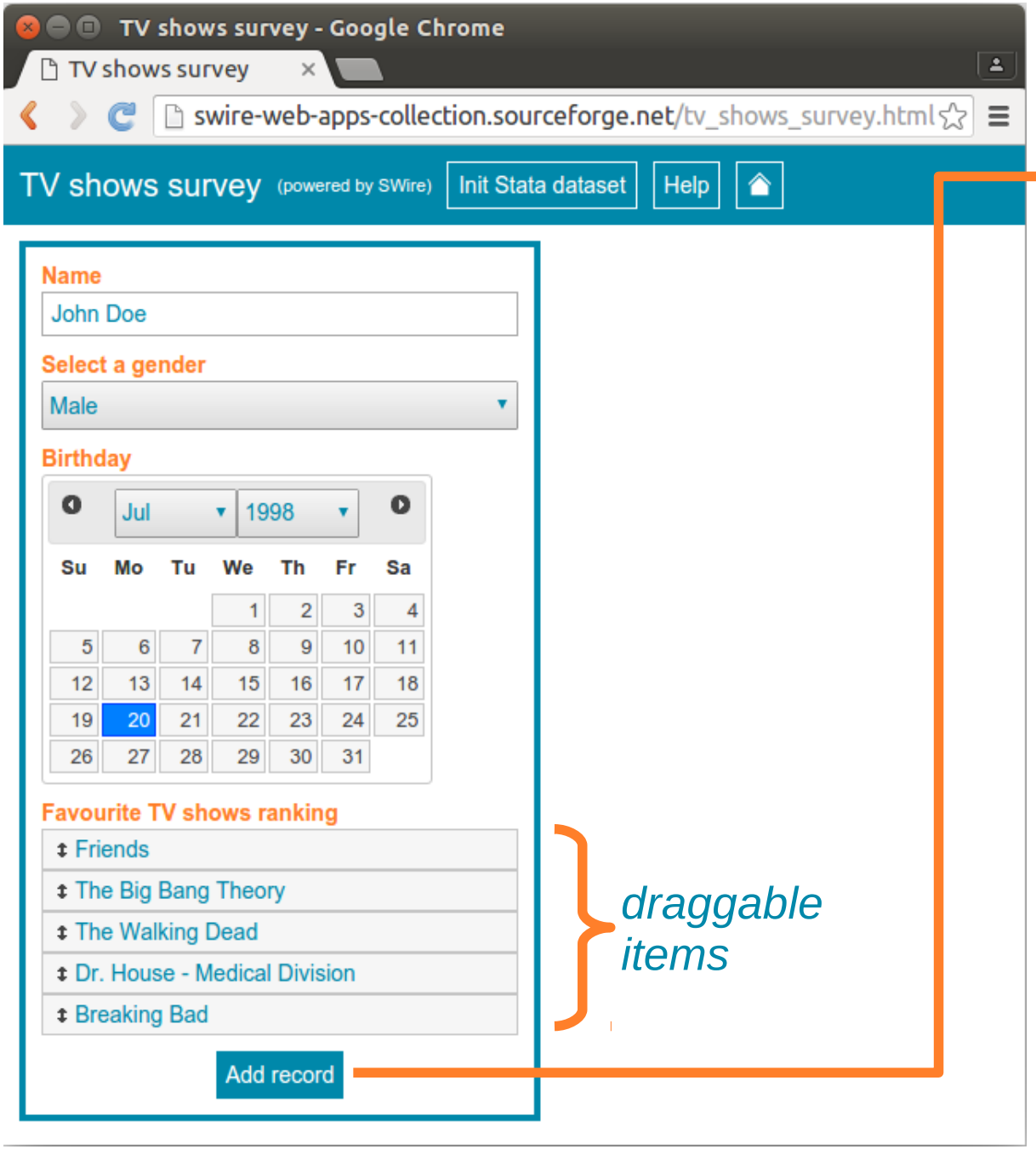
draggable marker

a row for each right-click on the marker

Stata dataset

	latitude	longitude	address
1.	41.900665	12.497321	Via Principe Amedeo, 4, 00184 Roma, Italy
2.	41.899555	12.498501	Via Cavour, 48, 00185 Roma, Italy
3.	41.896425	12.475166	Corso Vittorio Emanuele II, 120, 00186 Roma, Italy
4.	41.890579	12.512116	Via Statilia, 19, 00185 Roma, Italy
5.	41.9077	12.524647	Via Tiburtina, 269, 00162 Roma, Italy

TV shows survey



Added record in Stata dataset

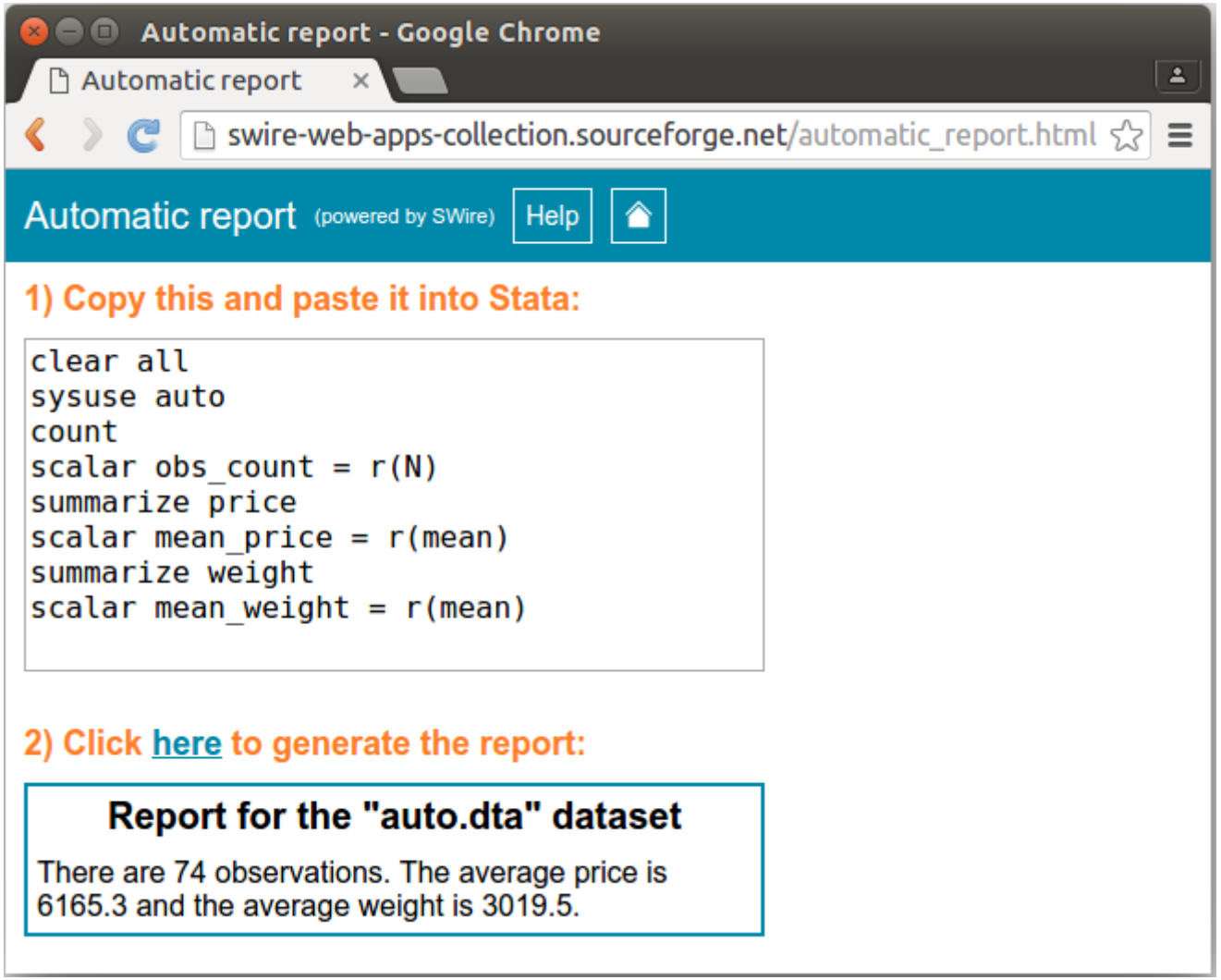
1.

name	gender
John Doe	**1**
birth_year **1998**	birth_month **7**
birth_day **20**	tv_show_1 **3**
tv_show_2 **1**	tv_show_3 **5**
tv_show_4 **2**	tv_show_5 **4**

Favourite TV shows codes

1) The Big Bang Theory
2) Dr. House - Medical division
3) Friends
4) Breaking Bad
5) The Walking Dead

Automatic report



SWire4QGIS

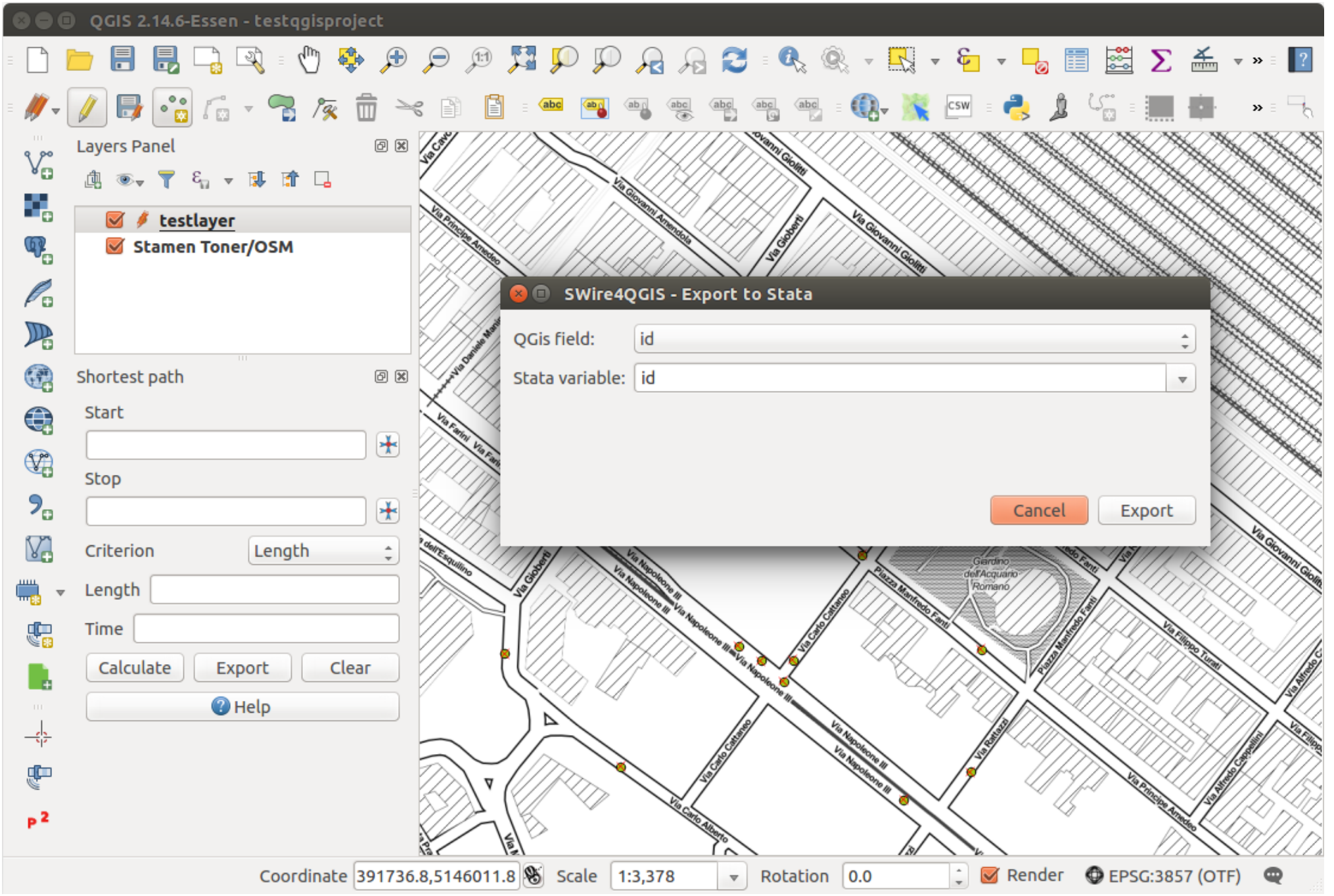
A QGIS plugin for exchanging numeric data with Stata

- powered by SWire
- written in C++
- the binary is available for Ubuntu
- source code is available

Project web page

https://sourceforge.net/projects/swire4qgis/

SWire4QGIS


QGIS 2.14.6-Essen - testqgisproject
Layers Panel
testlayer
Stamen Toner/OSM
Shortest path
Start
Stop
Criterion
Length
Length
Time
Calculate
Export
Clear
Help
SWire4QGIS - Export to Stata
QGis field:
id
Stata variable:
id
Cancel
Export
Coordinate
391736.8,5146011.8
Scale
1:3,378
Rotation
0.0
Render
EPSG:3857 (OTF)

Stata-Java API

An example:

The below call to the Stata-Java API:

```
com.stata.sfi.Data.addVarDouble("myvar");
```

… is equivalent to this Stata command:

```
generate double myvar = .
```

Full documentation:

http://www.stata.com/java/api/

SWire methods

Stata-Java API methods:

- com.stata.sfi.Data.addVarDouble
- com.stata.sfi.Data.dropVar
- com.stata.sfi.Data.renameVar
- com.stata.sfi.Matrix.getMatrix
- com.stata.sfi.Scalar.getValue
- com.stata.sfi.SFIToolkit.display
- etc...

Special methods:

- $appendRow
- $generateVars
- $getVarNames
- etc...

Note: special methods begin with “$”

SWire4R

A R package for exchanging data with Stata

- powered by SWire
- written in C++
- installation is documented for Ubuntu and Mac
- source code is available

Project web page

https://sourceforge.net/projects/swire4r/

SWire4R: an example session

Session in Stata:

```
. sysuse auto
```

Session in R:

```
> library(SWire4R)
> swire.getNumericVar("price")
$status
[1] 0

$data
 [1]  4099  4749  3799  4816  7827  5788  4453  5189 10372  4082 11385 14500
[13] 15906  3299  5705  4504  5104  3667  3955  3984  4010  5886  6342  4389
[25]  4187 11497 13594 13466  3829  5379  6165  4516  6303  3291  8814  5172
[37]  4733  4890  4181  4195 10371  4647  4425  4482  6486  4060  5798  4934
[49]  5222  4723  4424  4172  9690  6295  9735  6229  4589  5079  8129  4296
[61]  5799  4499  3995 12990  3895  3798  5899  3748  5719  7140  5397  4697
[73]  6850 11995
```

Note: status=0 means “no error”

News in SWire 0.2

SWire HTTPS server

$$f(x)$$

New special methods

Starting SWire in HTTP and HTTPS mode

HTTP

```
. swire start, http
```

HTTPS

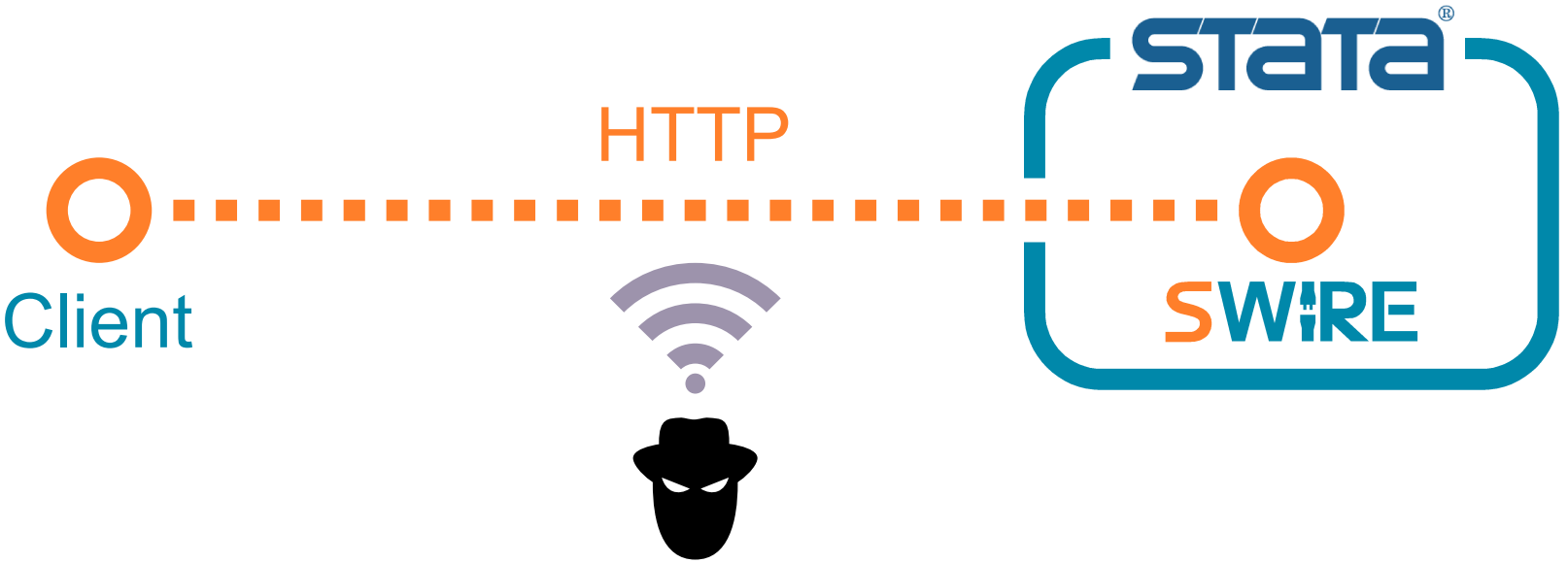
```
. swire start, https
```

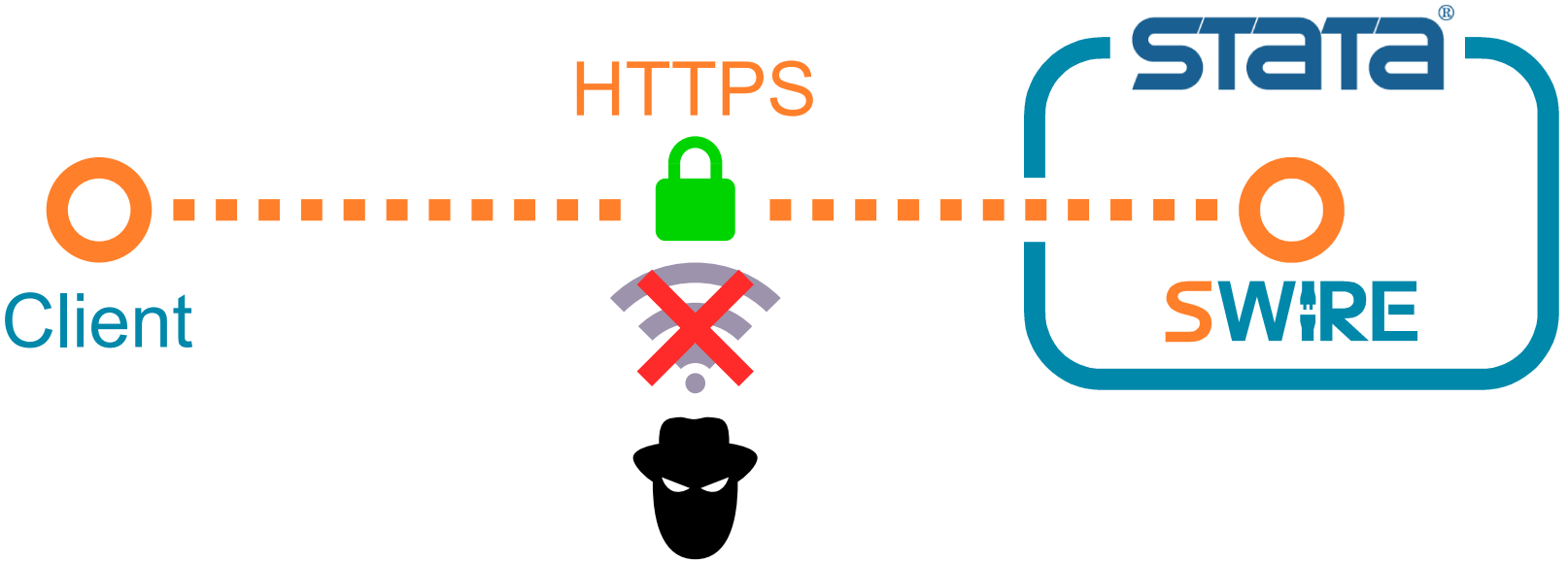
The HTTPS protocol

Privacy

Authentication

Privacy in HTTPS





Authentication in HTTPS

1) The server sends the public Certificate

2) The client trusts the server

Enable the SWire HTTPS server

1) create keys and certificates

2) export the SWire Certification Authority Certificate

3) trust the SWire Certification Authority Certificate

4) start the SWire HTTPS server

The SWire keystore

The SWire keystore contains:

- the private key
- the SWire Certification Authority self-signed Certificate
- the *localhost* Certificate for the SWire HTTPS server

Create the SWire keystore:

```
. swire initsecurity
```

Export the SWire Certification Authority Certificate

SWire keystore

export

SWire Certification Authority Certificate

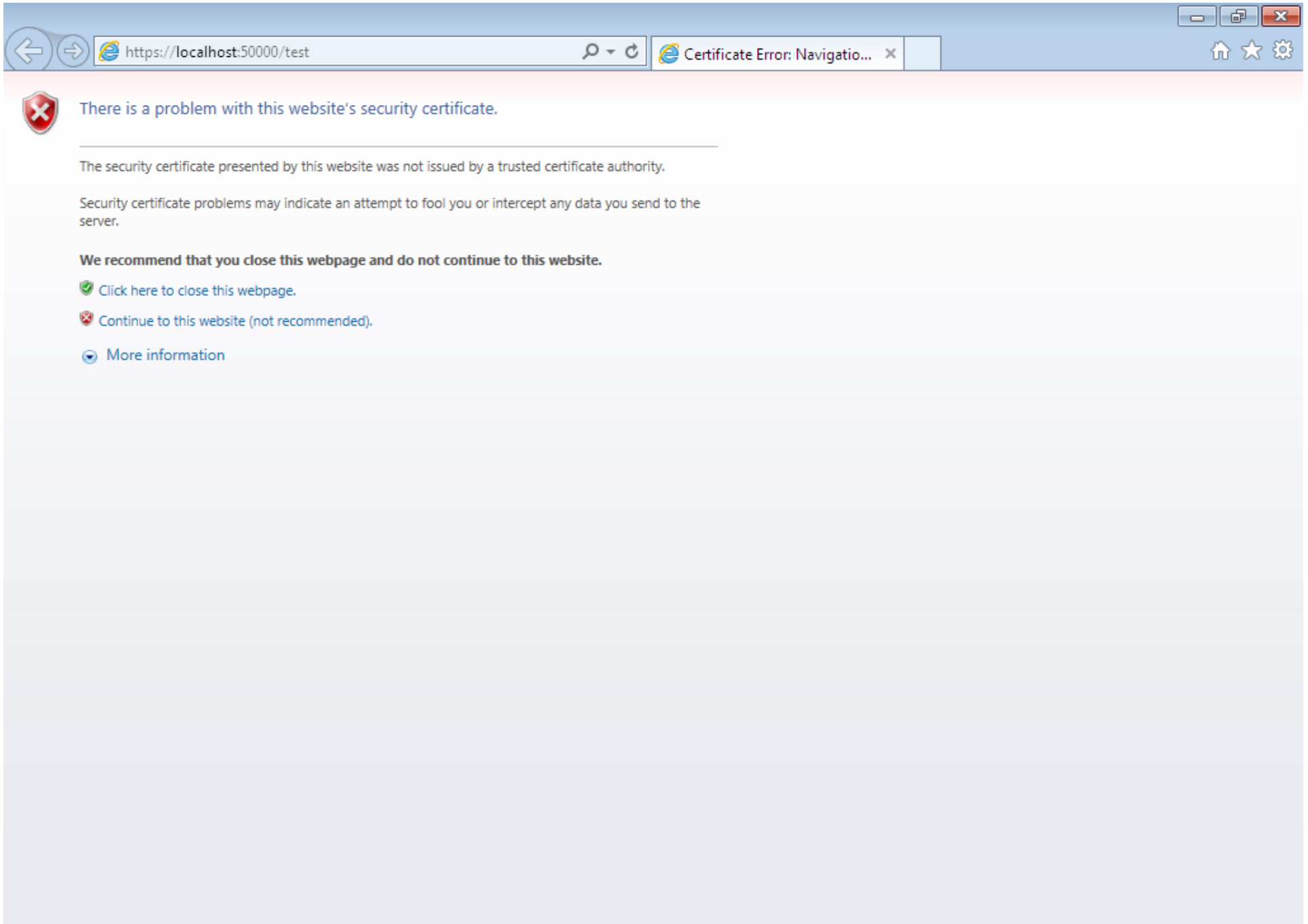
Export the Certificate:

```
. swire exportcert
```

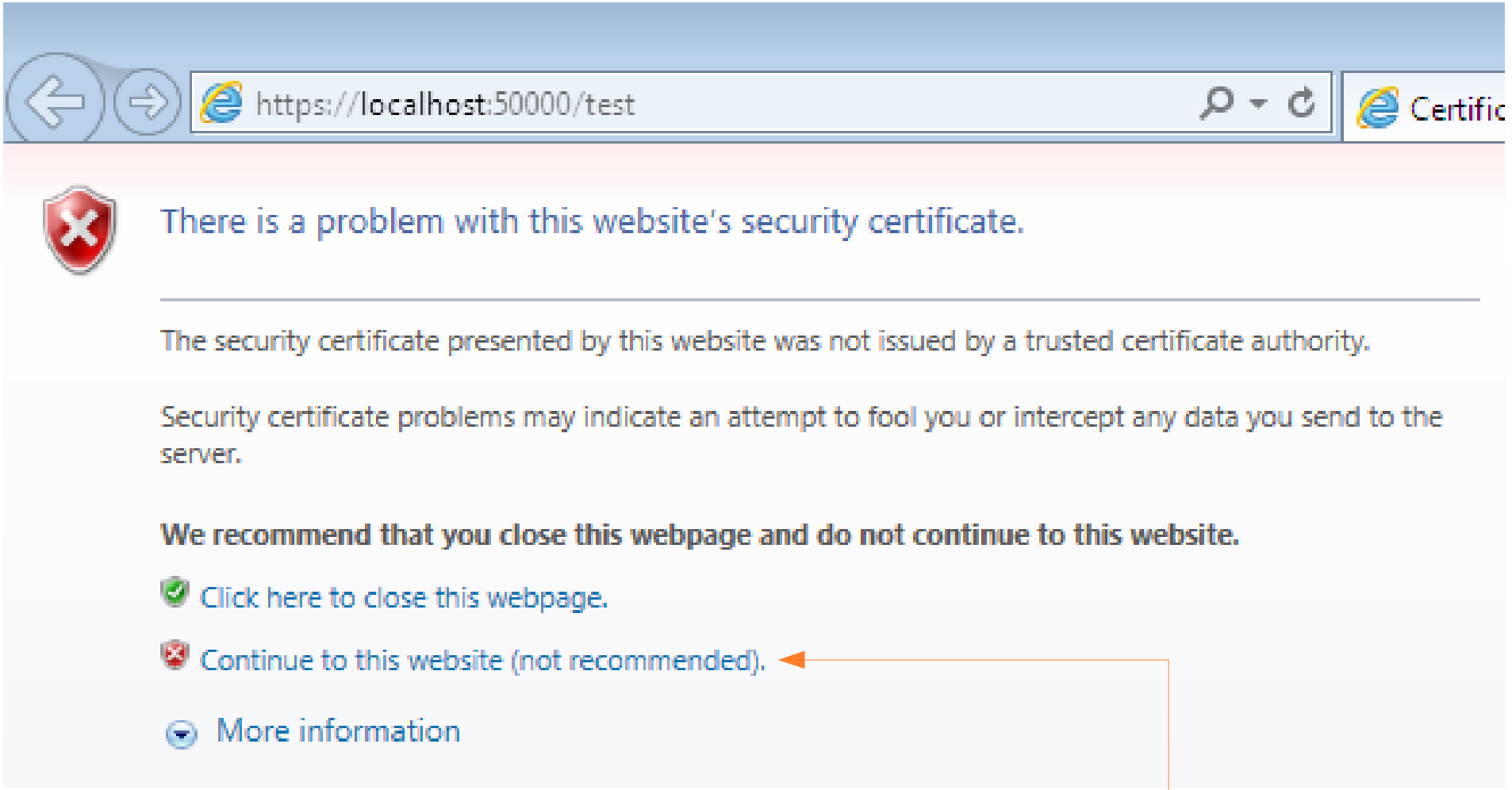
Starting SWire in HTTPS mode

```
. swire start, https
------------------------------------------------------------------------------
The SWire HTTPS server is listening on port 50000
------------------------------------------------------------------------------

Test page: https://localhost:50000/test (it may be necessary to force the
> browser being used to trust this page)
Certification chain expires on: Mon Jun 06 22:33:58 CEST 2022

Warning:

SWire will accept every incoming connection on the port on which the serve
> r is listening. Therefore a web page that was opened whilst navigating o
> n the internet, or a malicious program running on your local network can
> silently modify or read your Stata data (dataset, scalars or macros).
```

SWire HTTPS test page: https://localhost:50000/test

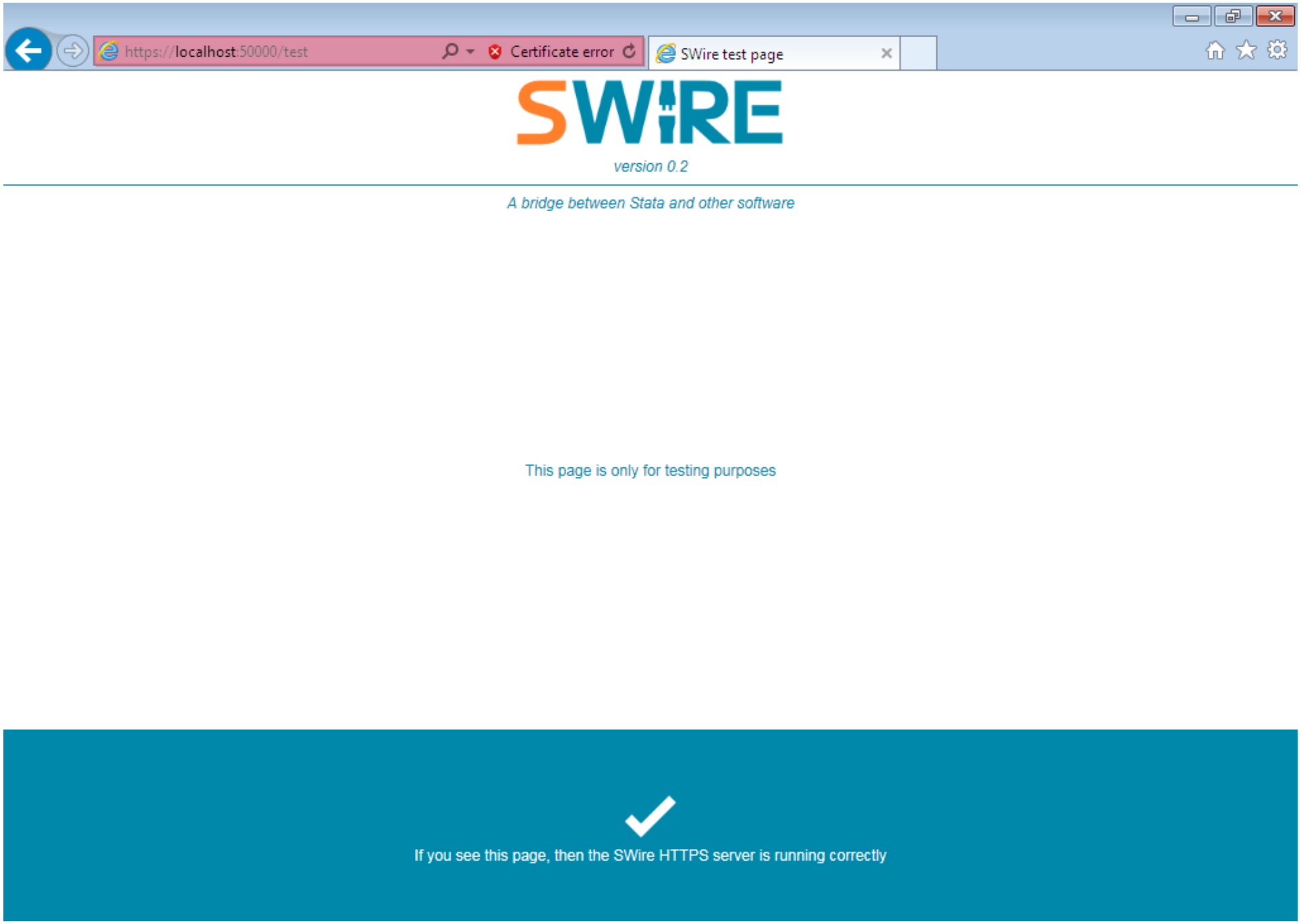


Trust the SWire HTTPS server

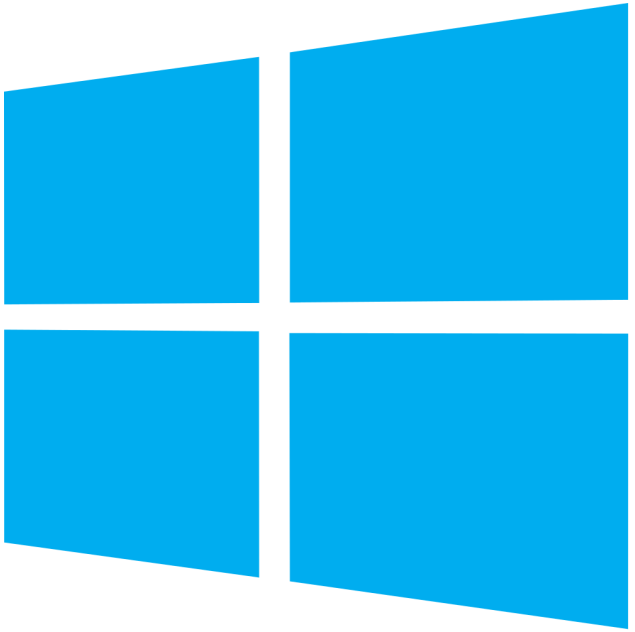


click here to trust the SWire HTTPS server

The test page is visible: the SWire HTTPS server is running correctly

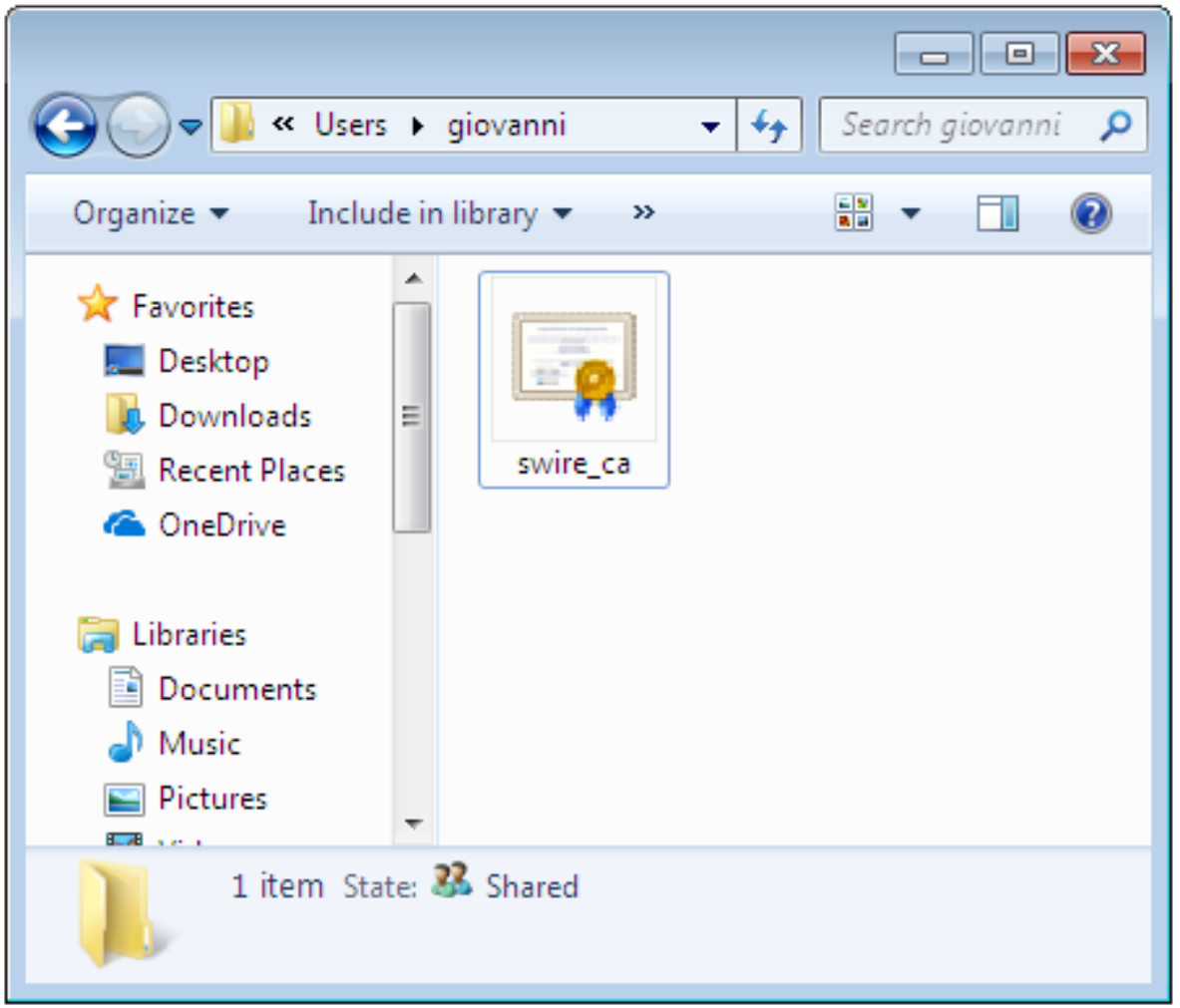


Trusting the Swire Certification Authority Certificate in Windows



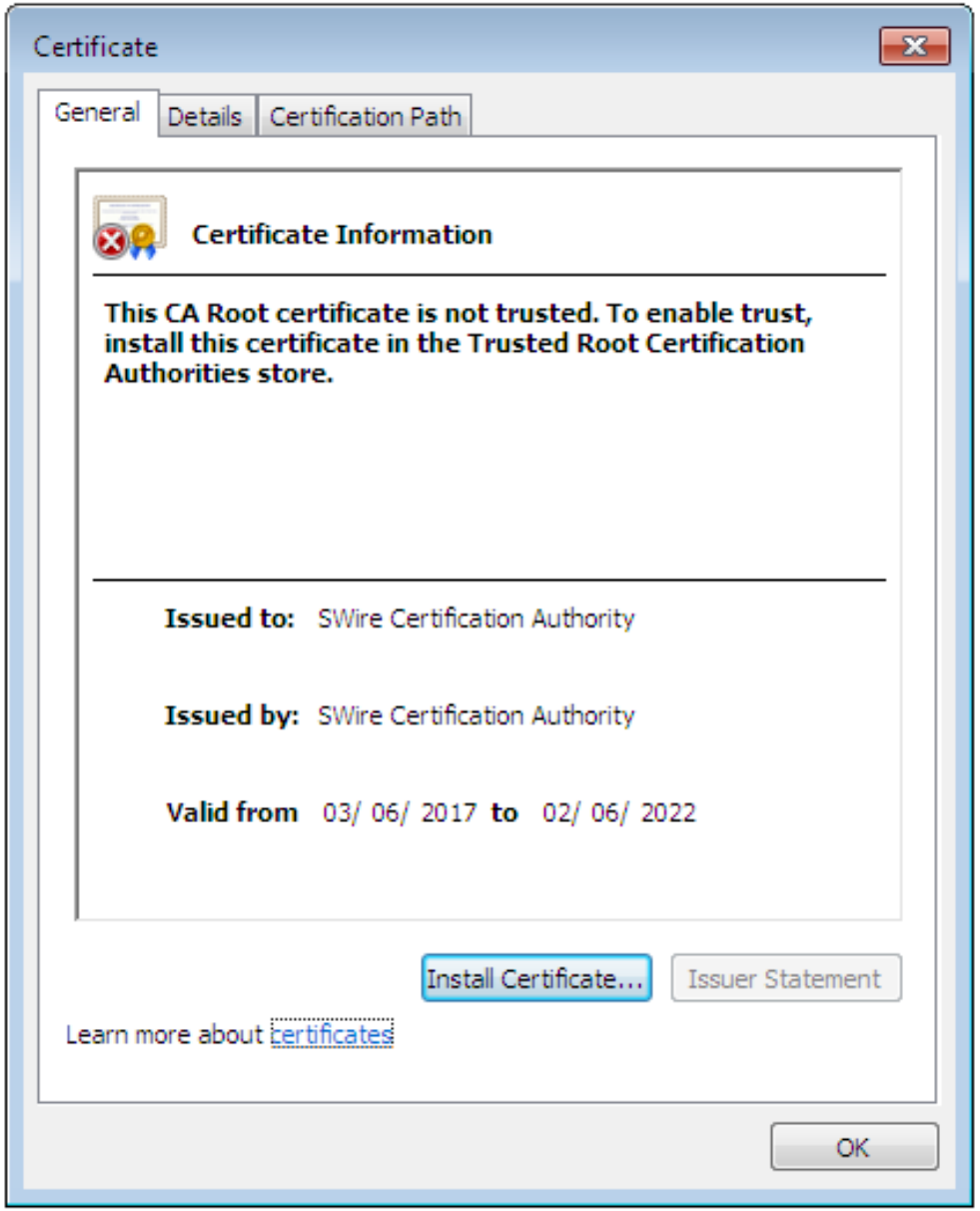
Trusting the Swire Certification Authority Certificate in Windows (1/3)

Double-click on the Certificate



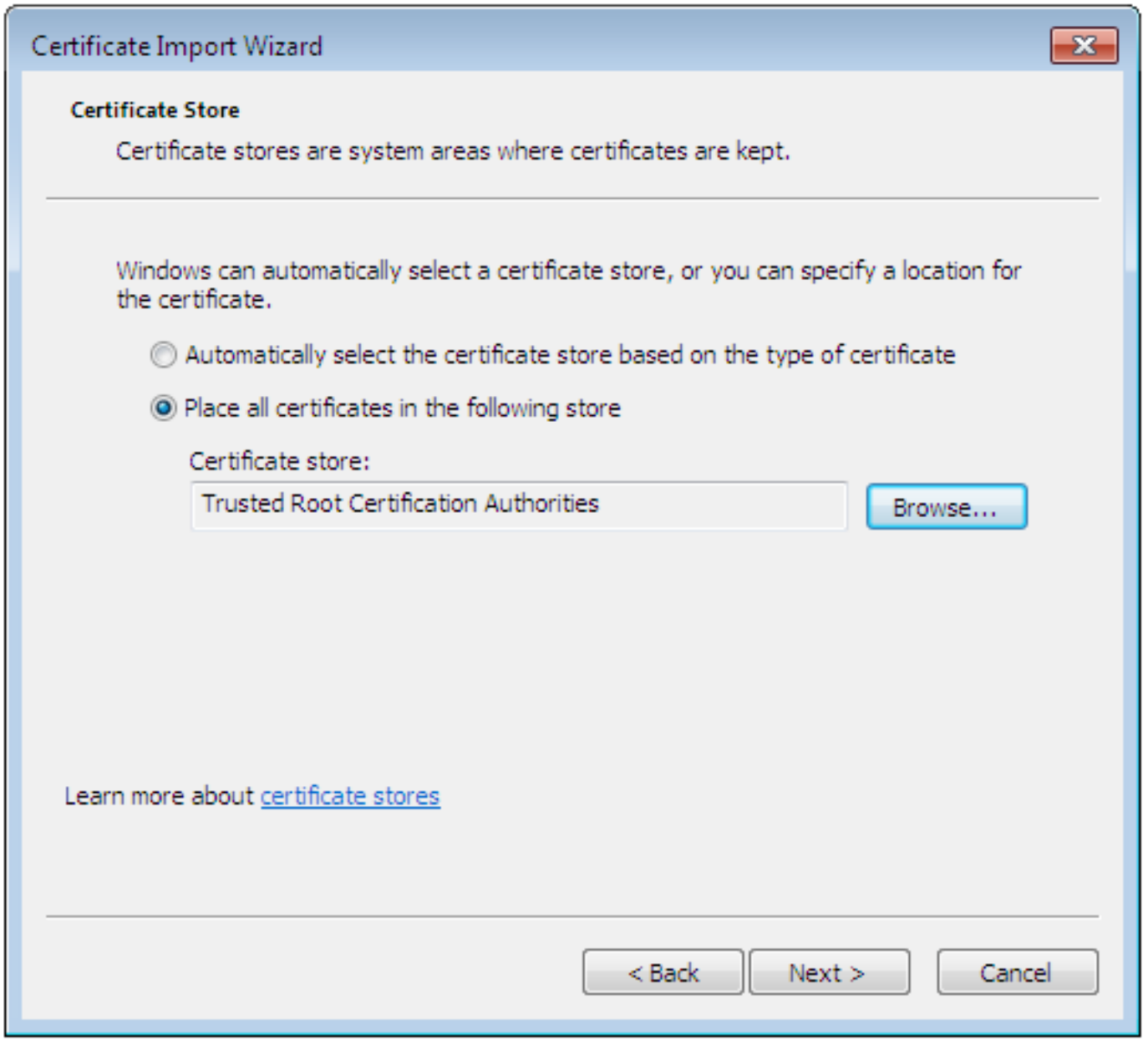
Trusting the Swire Certification Authority Certificate in Windows (2/3)

Click on “Install Certificate...”



Trusting the Swire Certification Authority Certificate in Windows (3/3)

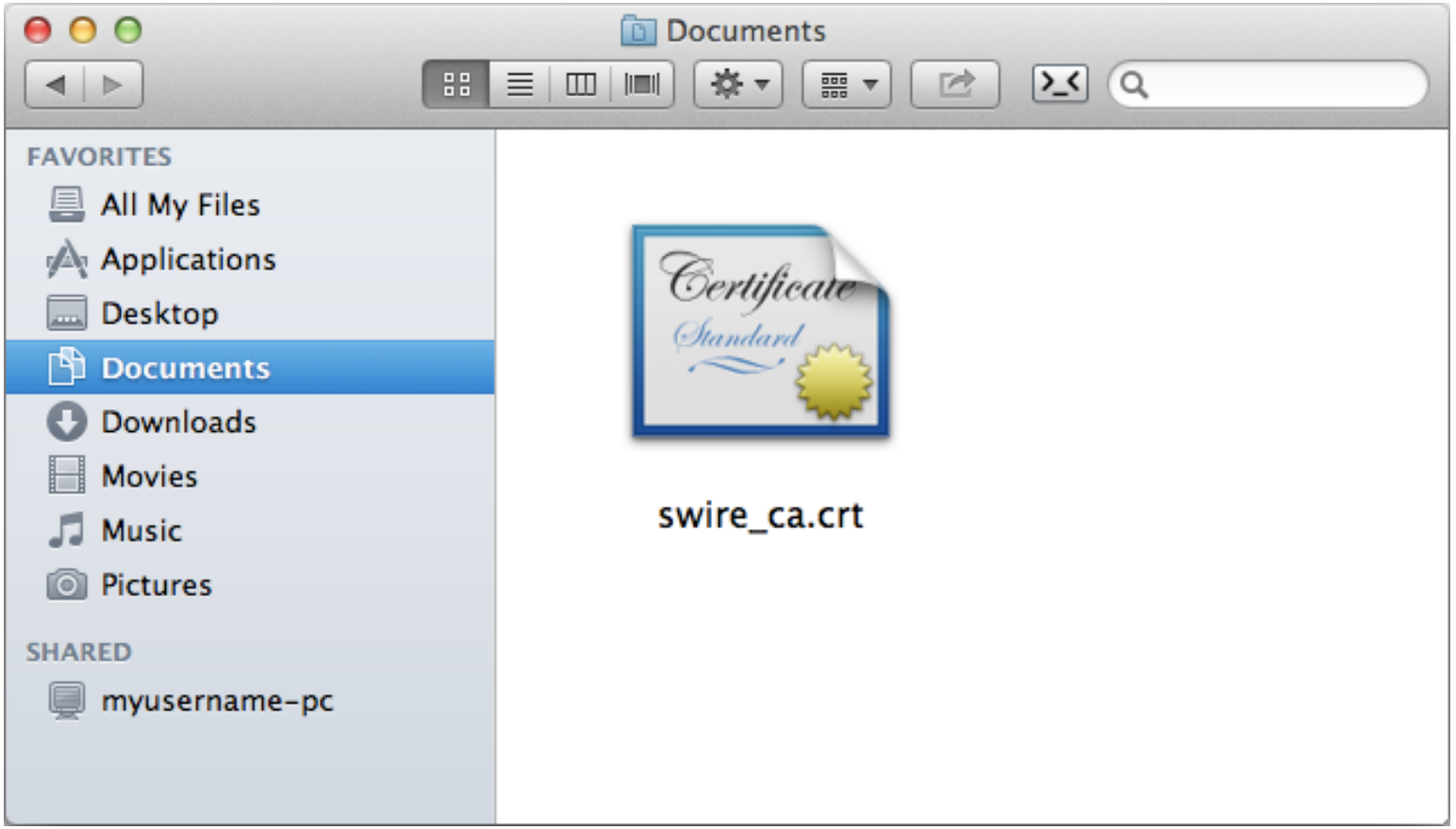
Place the Certificate in the “Trusted Root Certification Authorities” store



Trusting the Swire Certification Authority Certificate in Mac

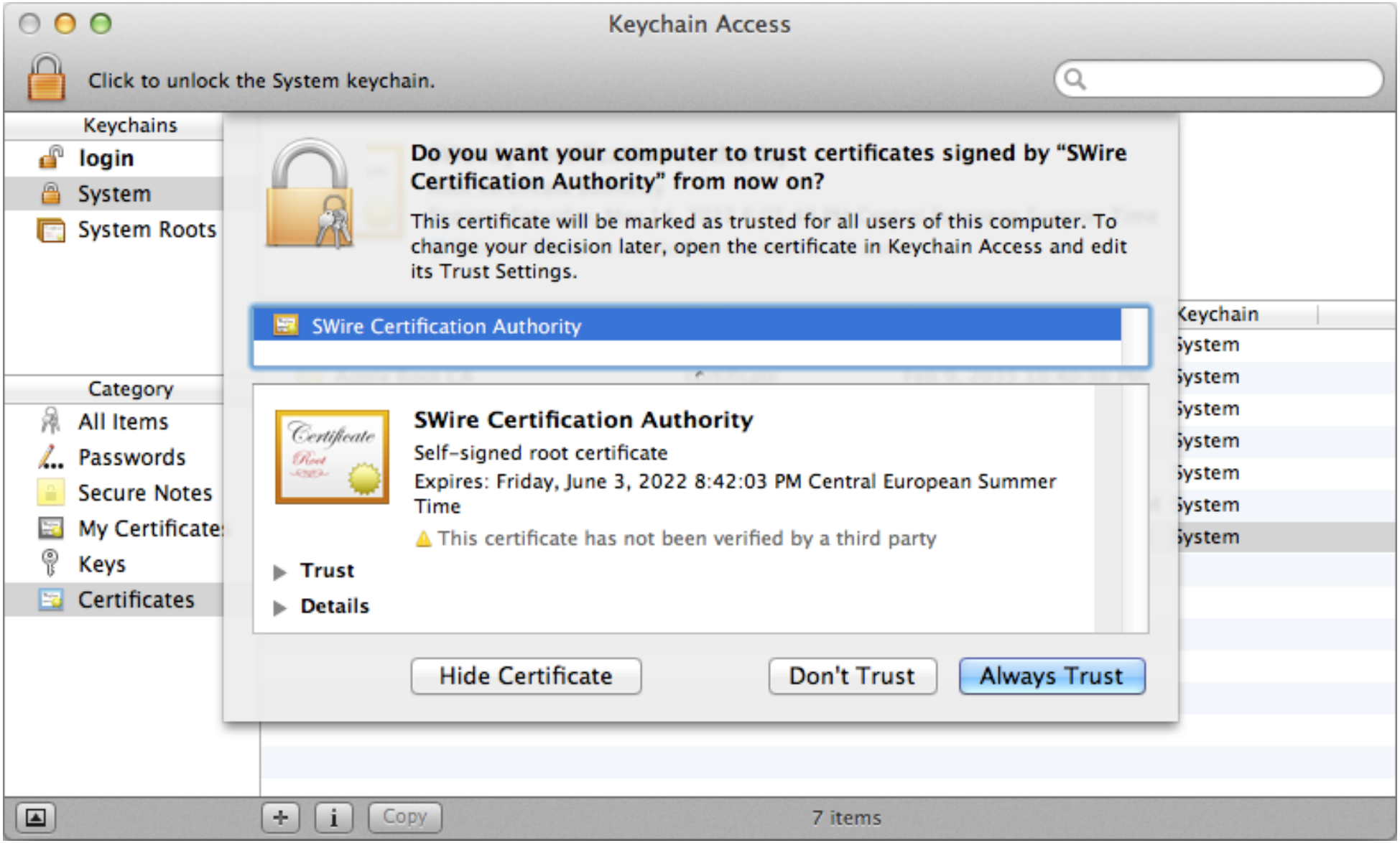
Trusting the Swire Certification Authority Certificate in Mac (1/4)

Double-click on the Certificate



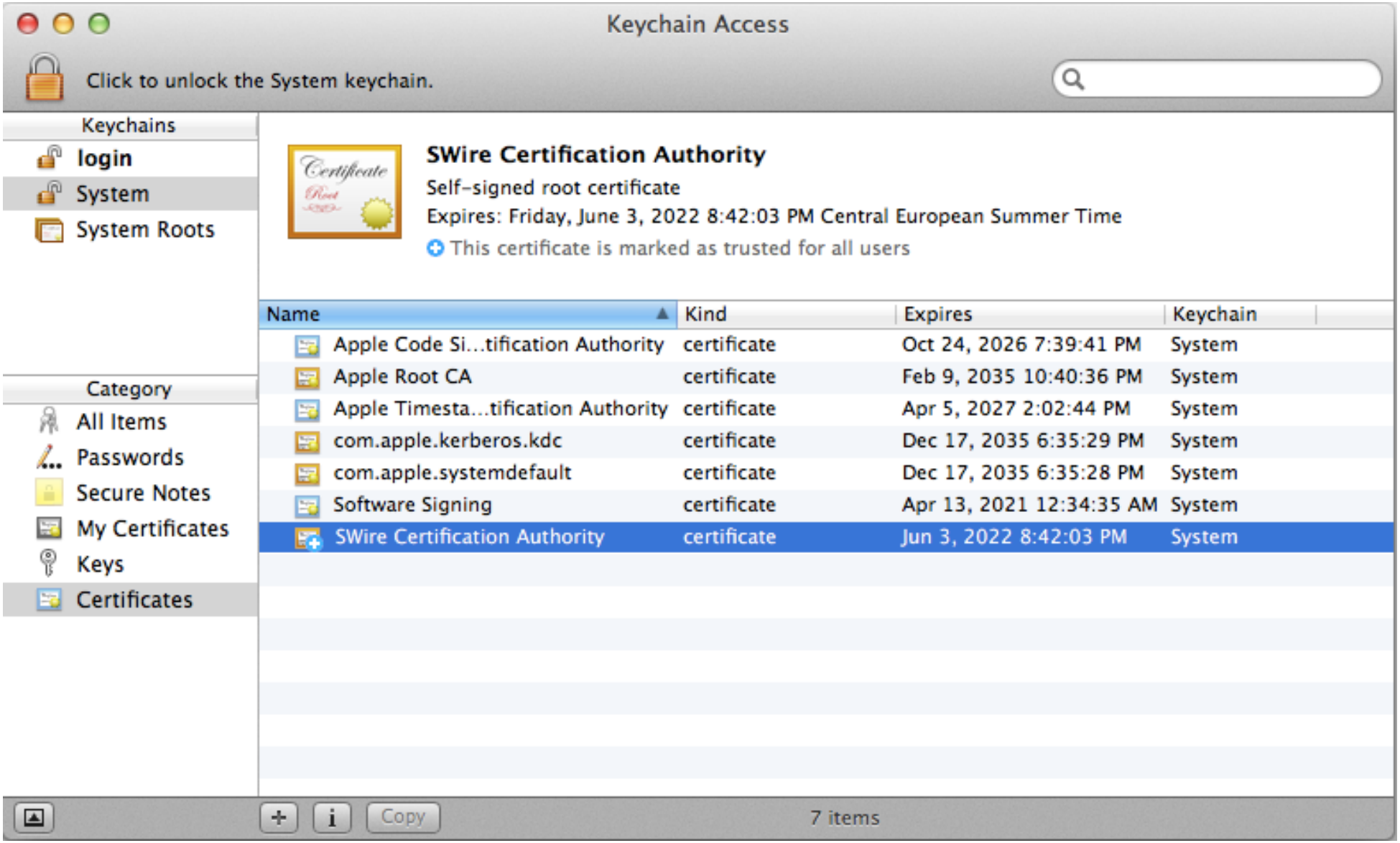
Trusting the Swire Certification Authority Certificate in Mac (2/4)

Click on “Always Trust”



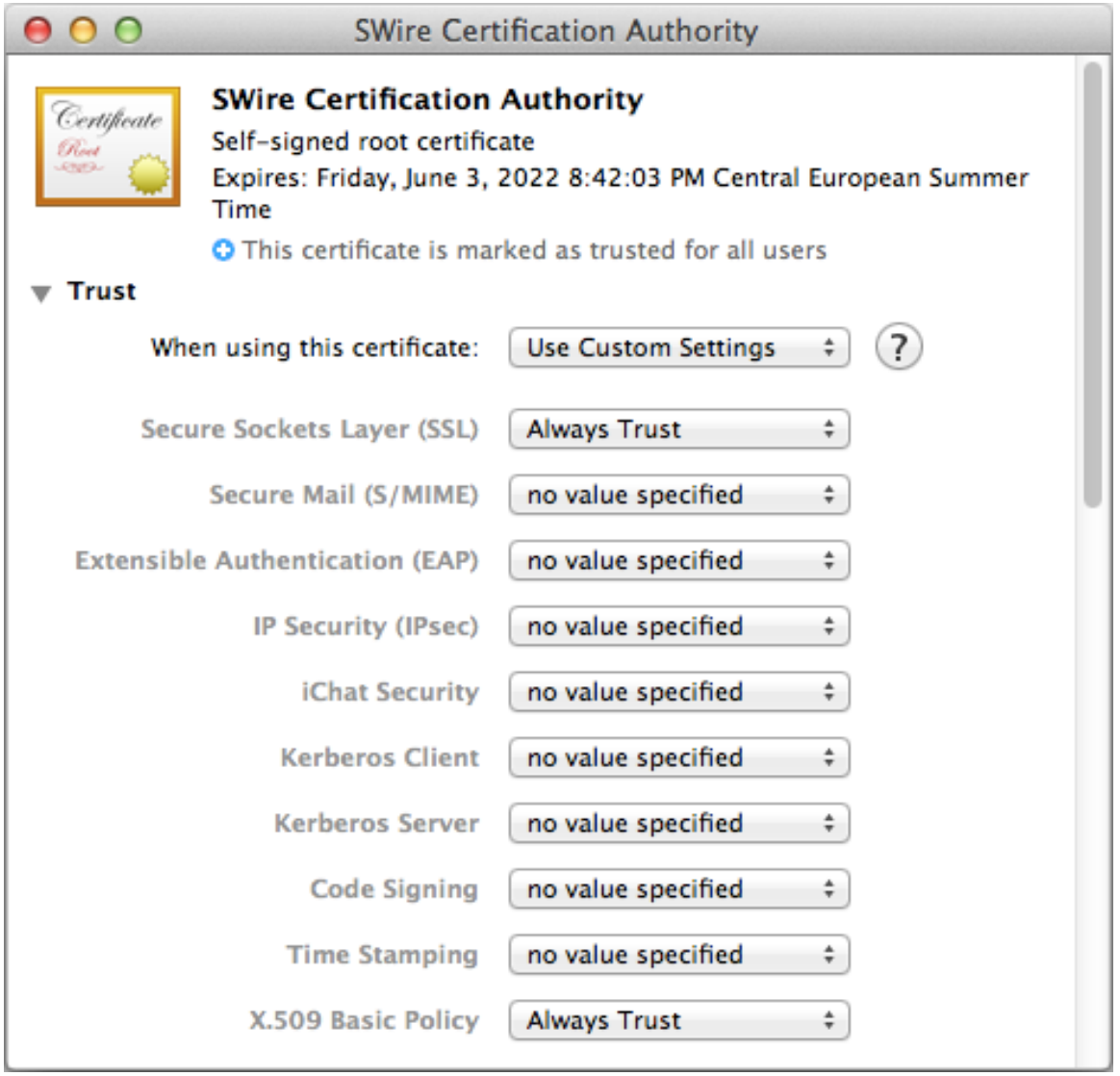
Trusting the Swire Certification Authority Certificate in Mac (3/4)

Right-click on “Swire Certification Authority” and select “Get Info”



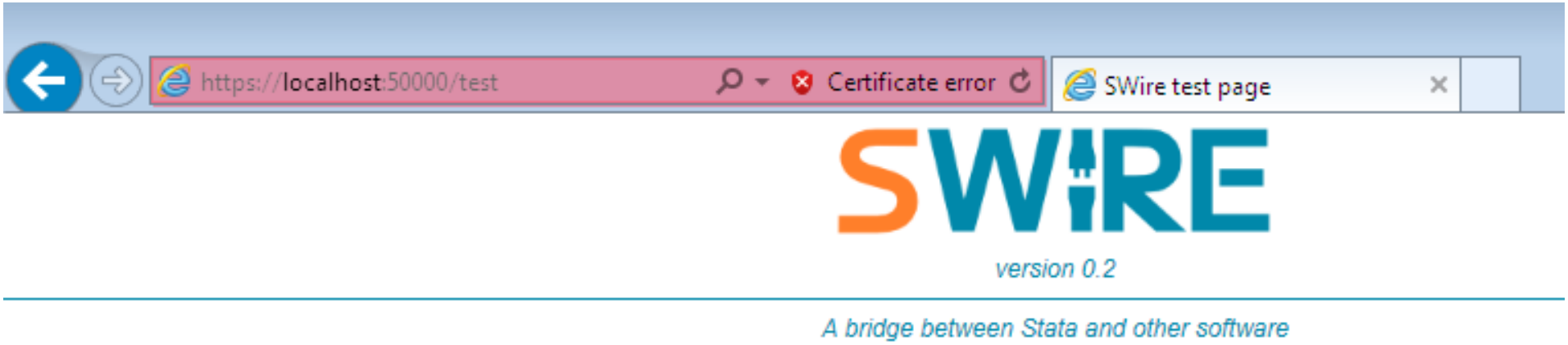
Trusting the Swire Certification Authority Certificate in Mac (4/4)

Always trust “Secure Sockets Layer (SSL)



Before and after the Certificate has been trusted by the OS

Before:



After:

Office Add-ins

Apps which extend Office by:

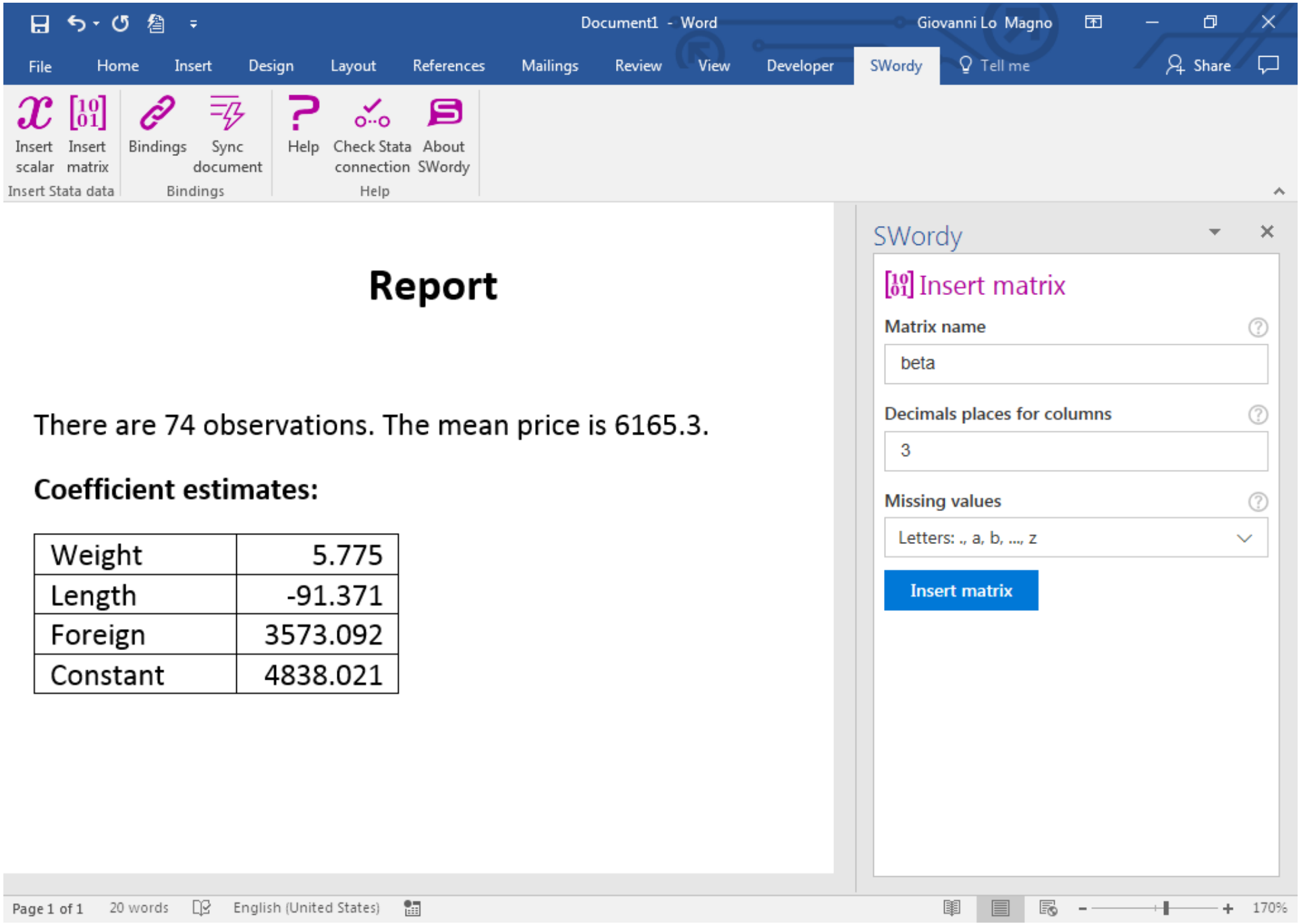
- adding new interaction with documents
- allowing the retrieving of external data
- exposing third-party functions
- … and a lot more

Office add-in architecture

Office add-in = **Manifest file** + **Web app on a HTTPS server**

SWordy: connecting Word to Stata

A first look at SWordy



SWordy's requirements

Software requirements:

Windows users:

≥ 13

≥ 0.2

≥ 2016

≥ 11

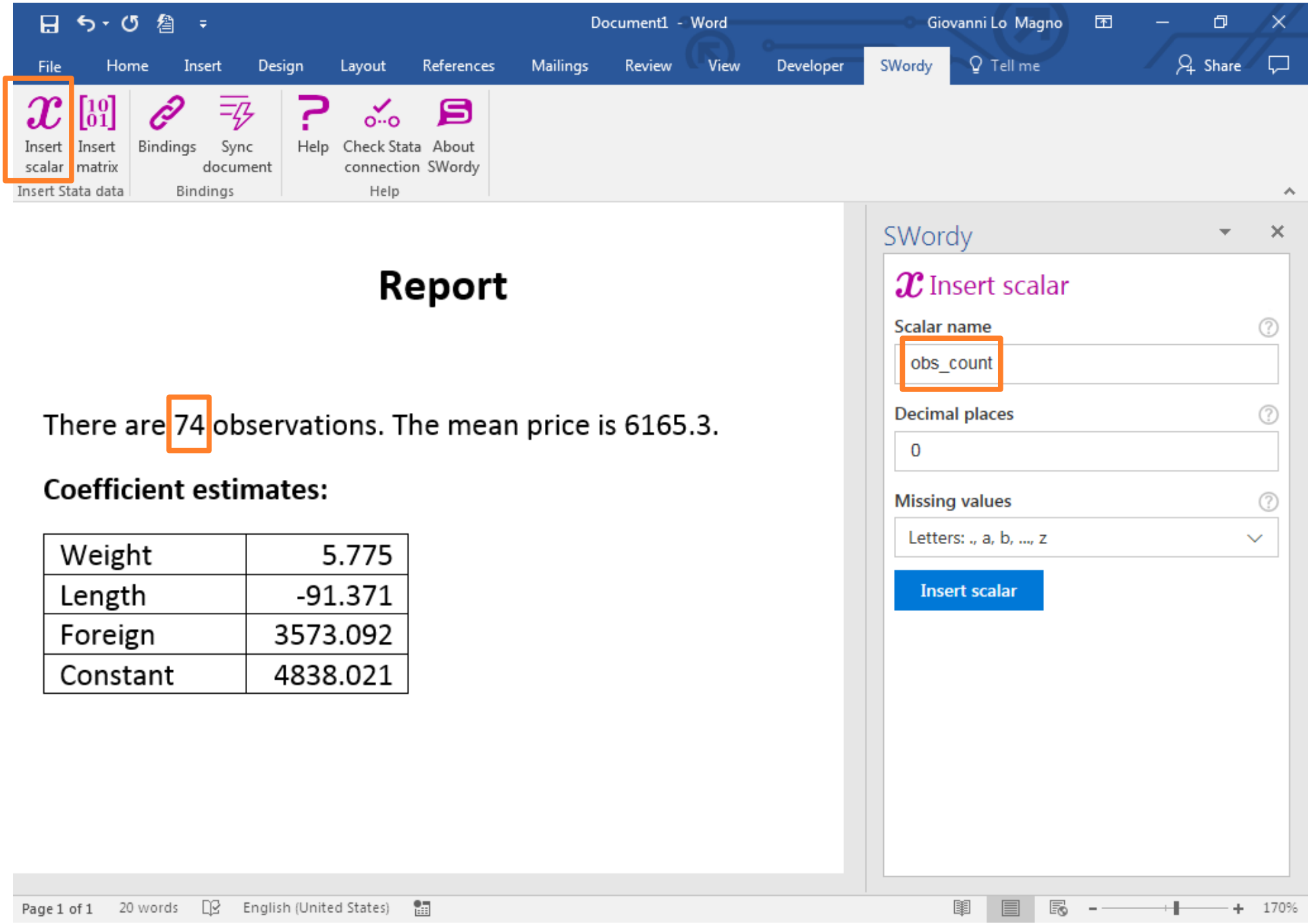
Other requirements:

- start the SWire HTTPS server (**`swire start, https`**)
- trust the SWire Certification Authority Certificate in your OS

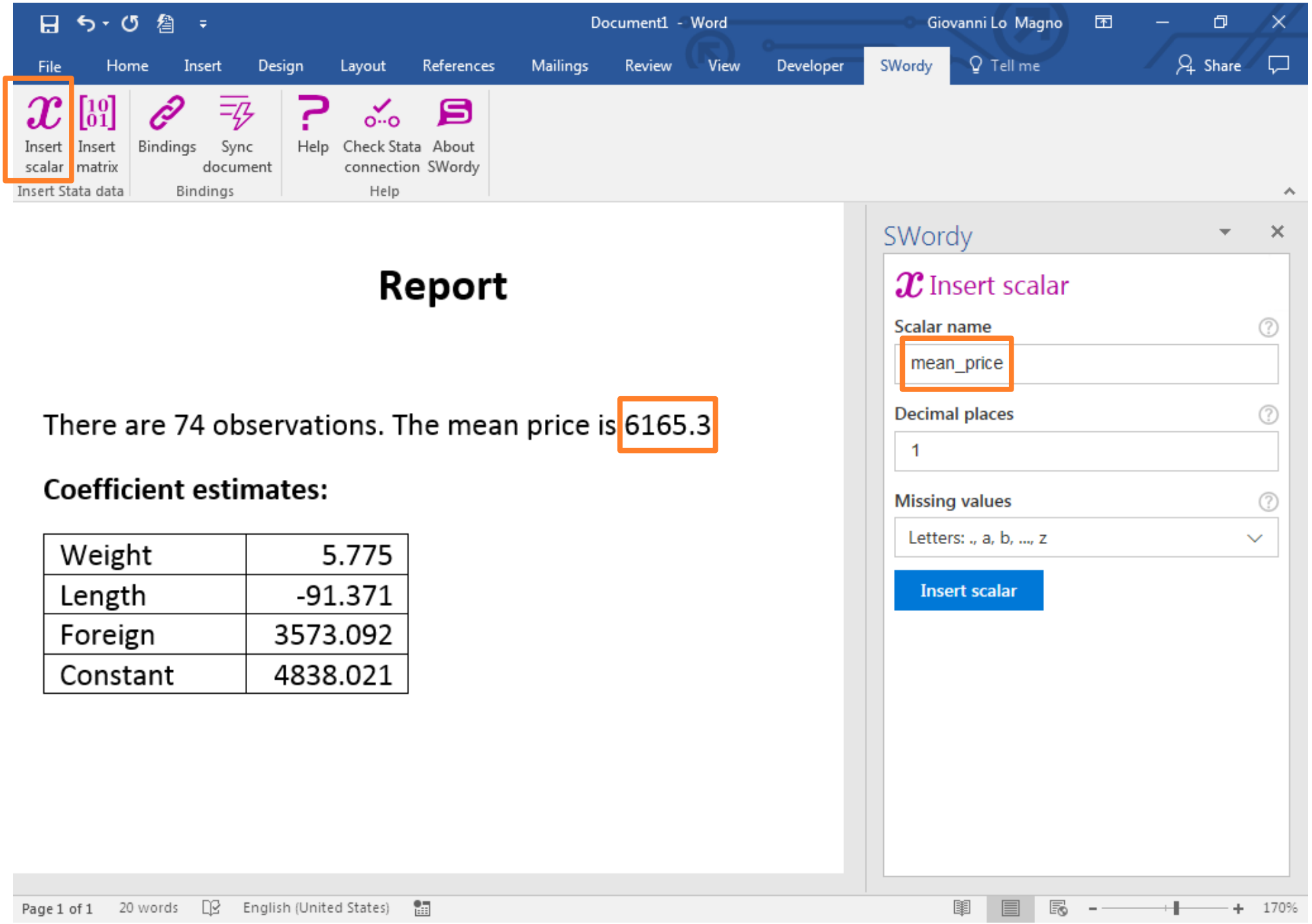
Preparing the data for the report

```
. sysuse auto
(1978 Automobile Data)

. count
  74

. scalar obs_count = r(N)

. summarize price

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         74    6165.257    2949.496       3291      15906

. scalar mean_price = r(mean)

. quietly regress price weight length foreign

. matrix beta = e(b)'
```

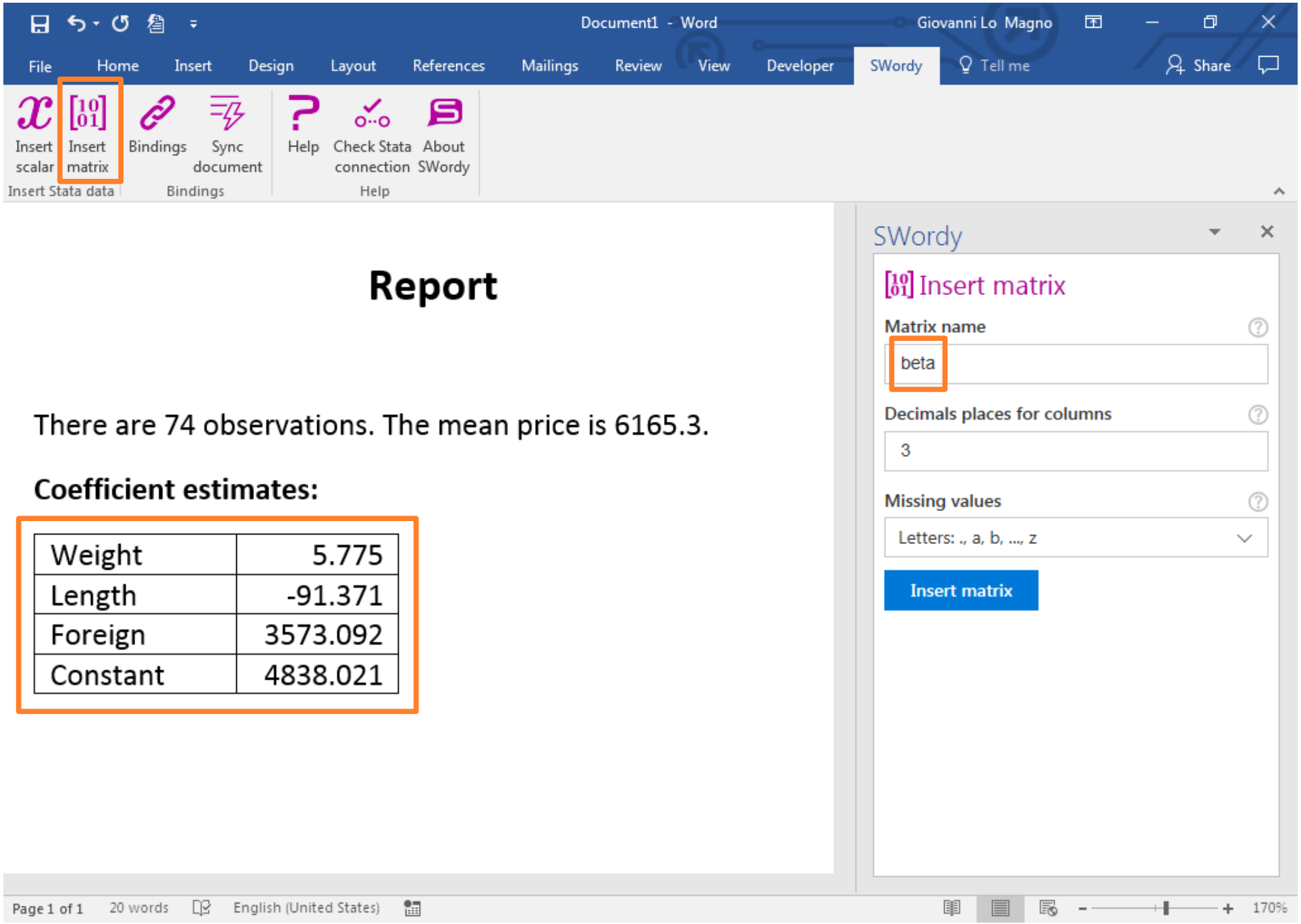
Inserting scalar data: number of observations



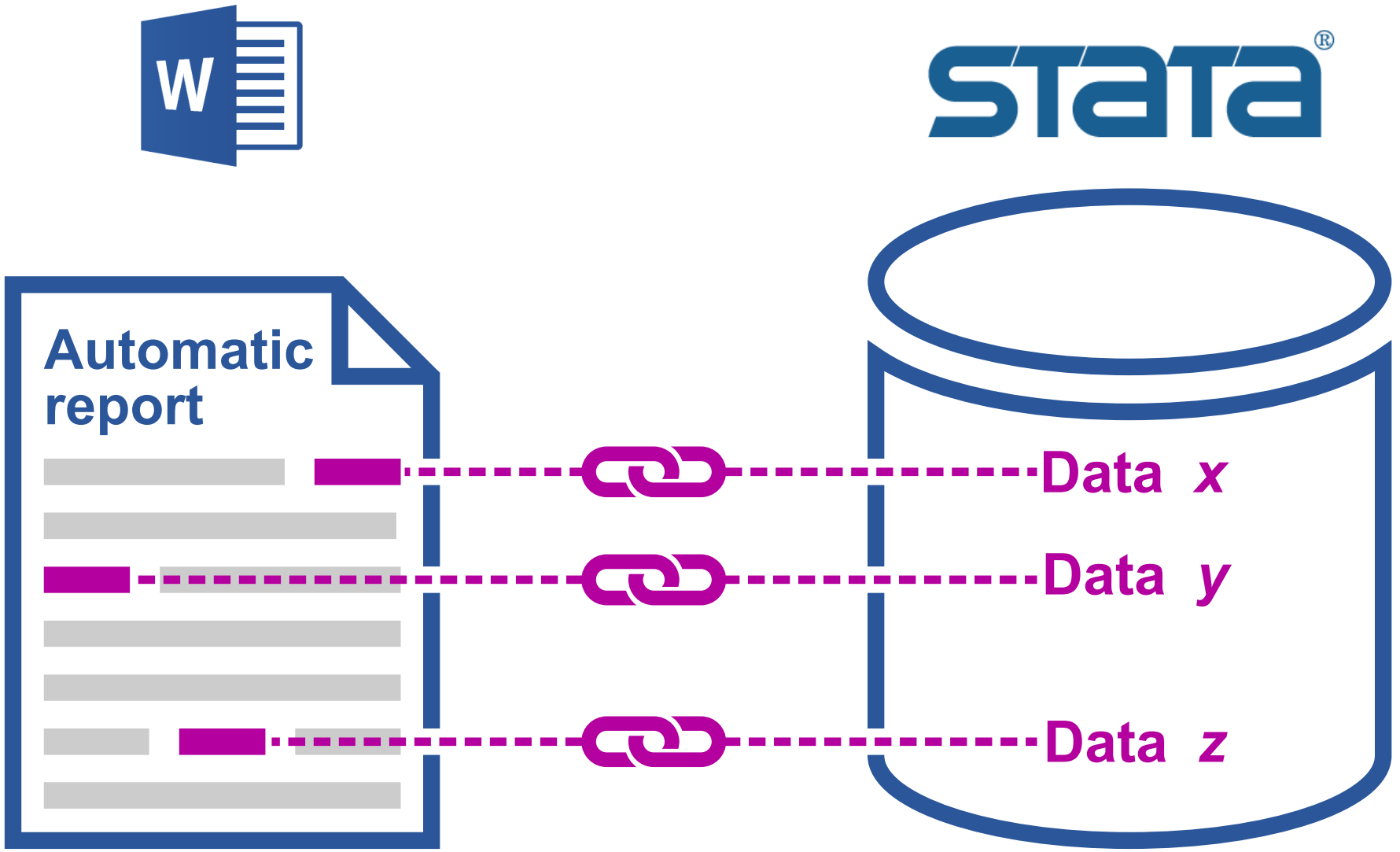
Inserting scalar data: mean price



Inserting matrix data: coefficient estimates



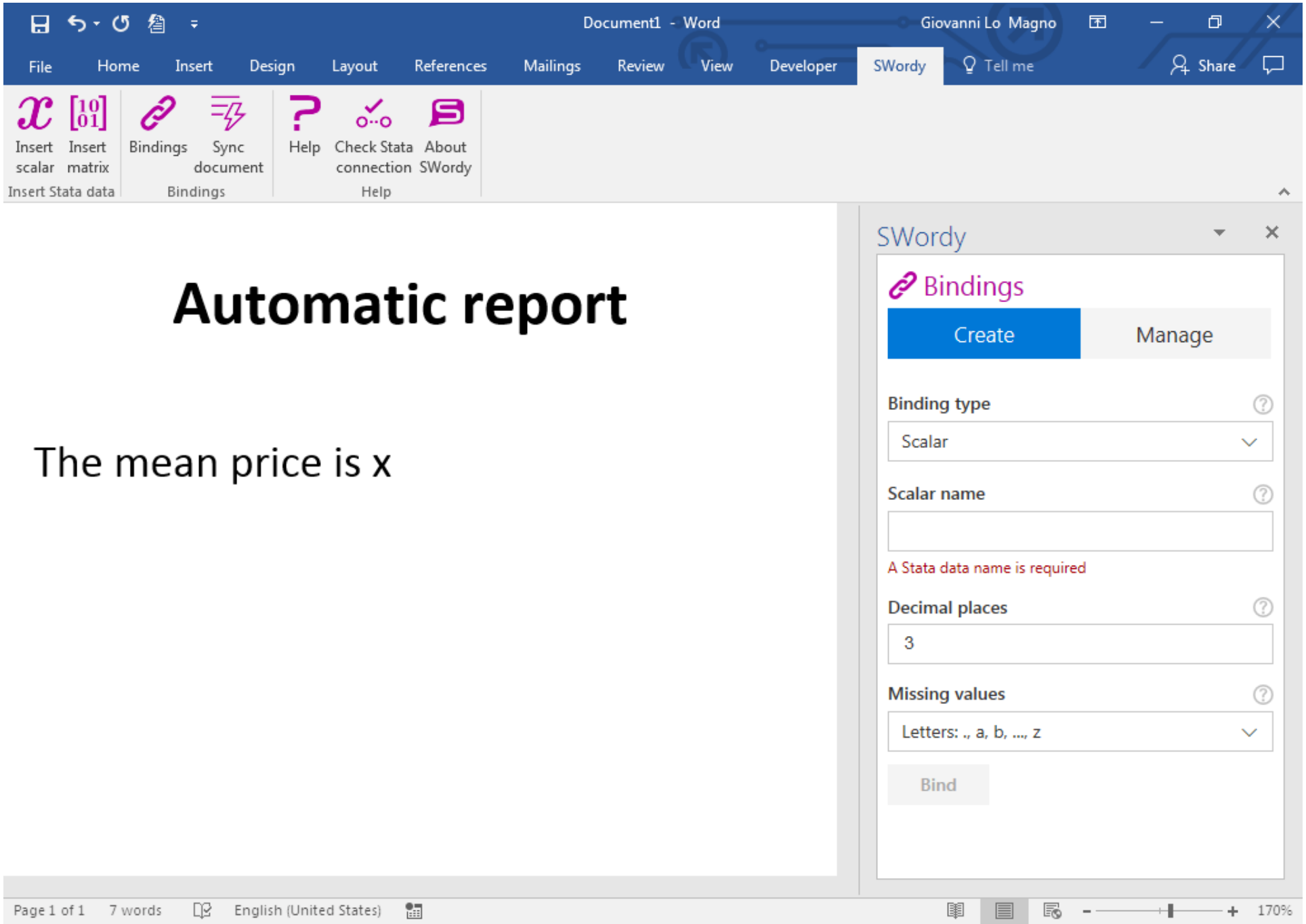
Bindings



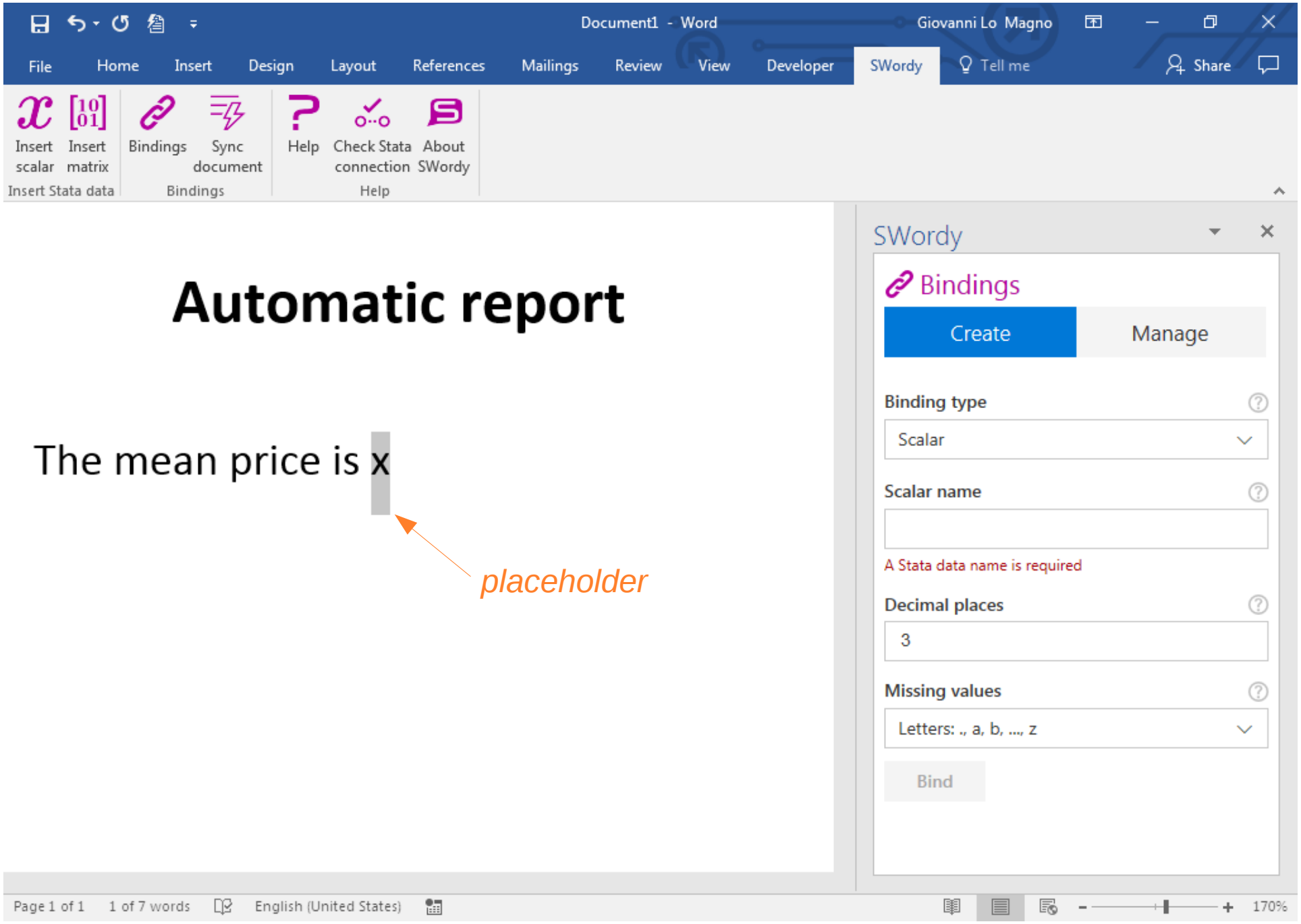
Binding to a Stata scalar – Step 0: optionally prepare the Stata scalar

```
. sysuse auto
(1978 Automobile Data)

. summarize price

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
       price |         74    6165.257    2949.496        3291      15906

. scalar mean_price = r(mean)

. display mean_price
6165.2568
```

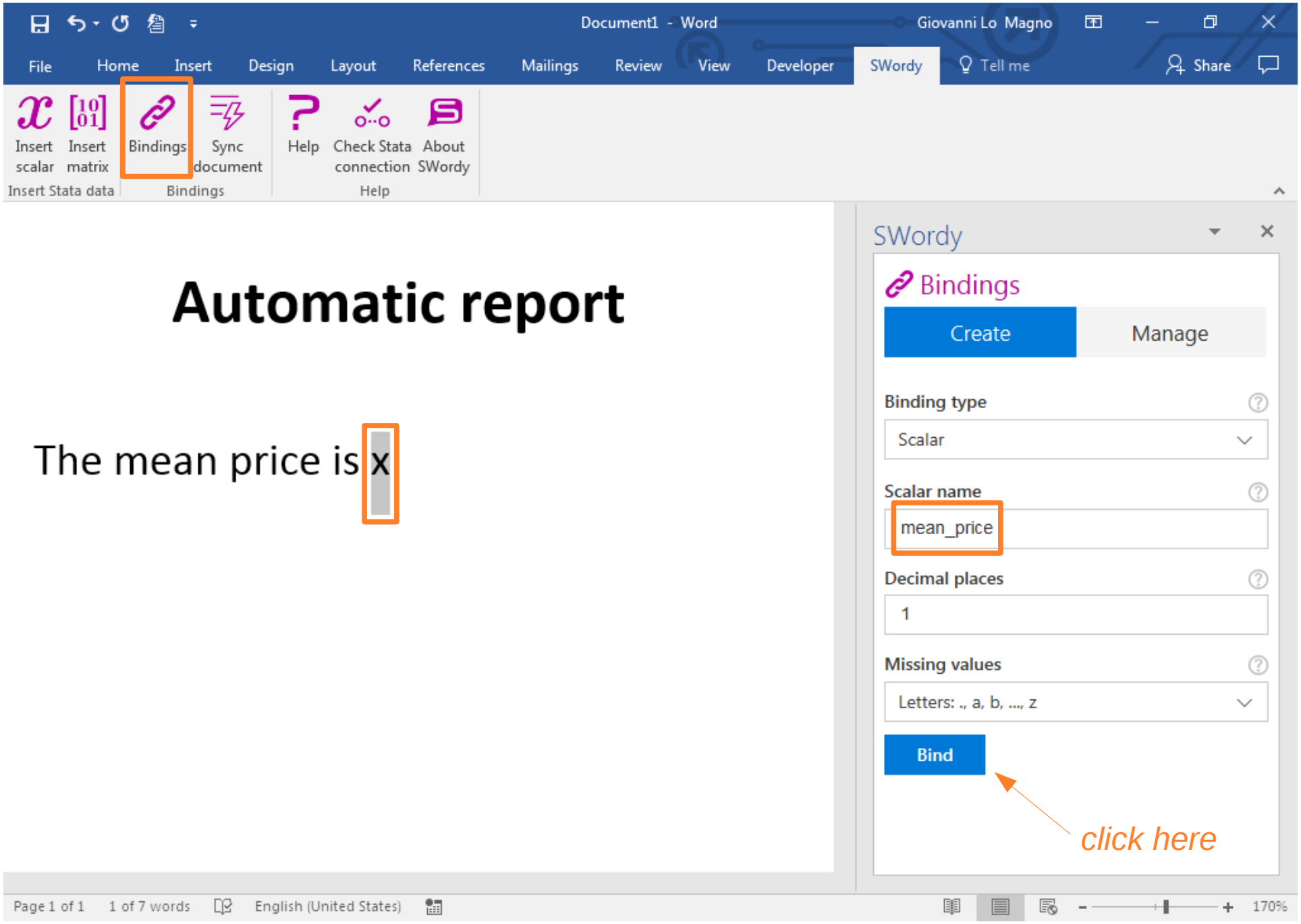
Binding to a Stata scalar – Step 1: write the text



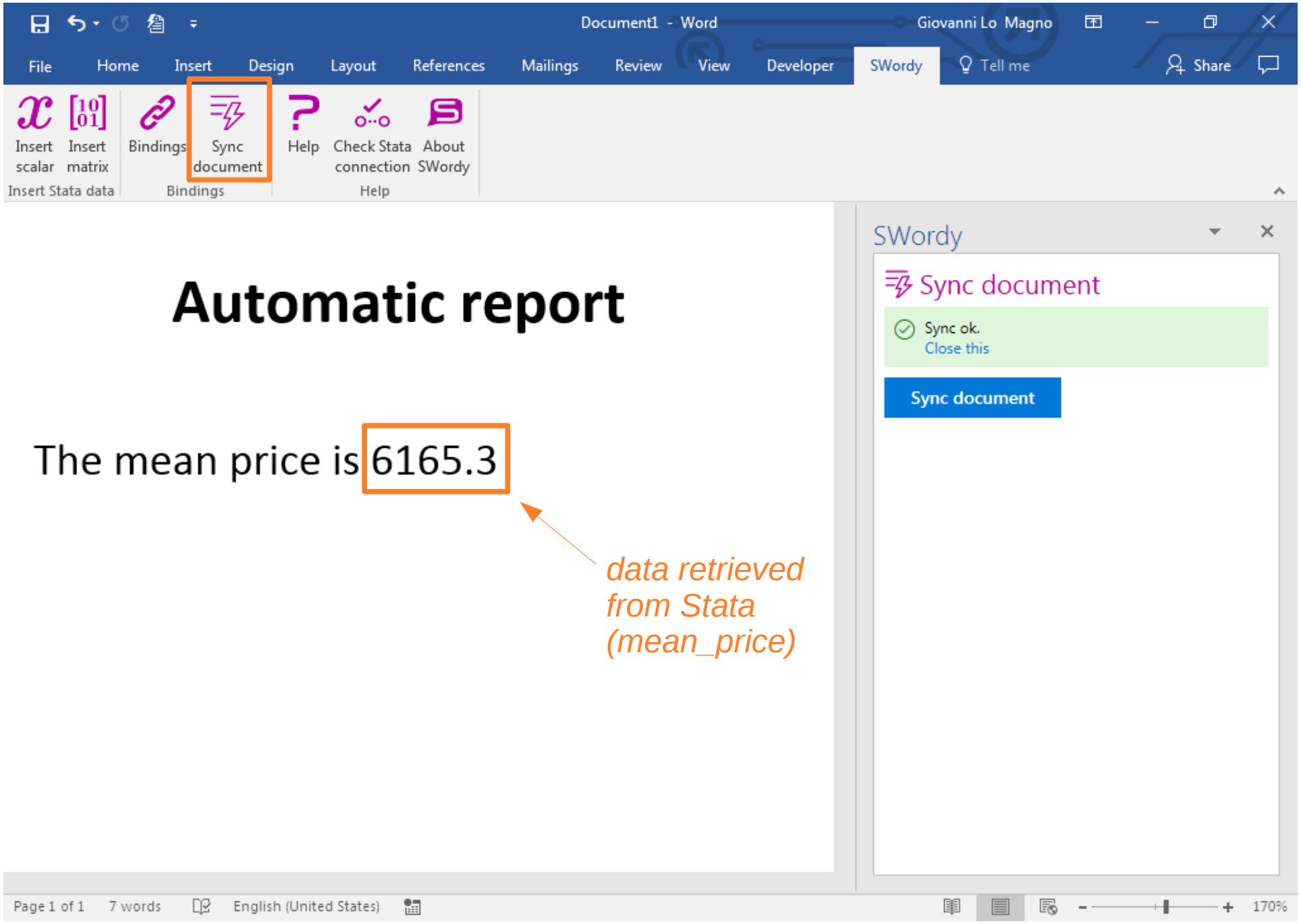
Binding to a Stata scalar – Step 2: select the placeholder



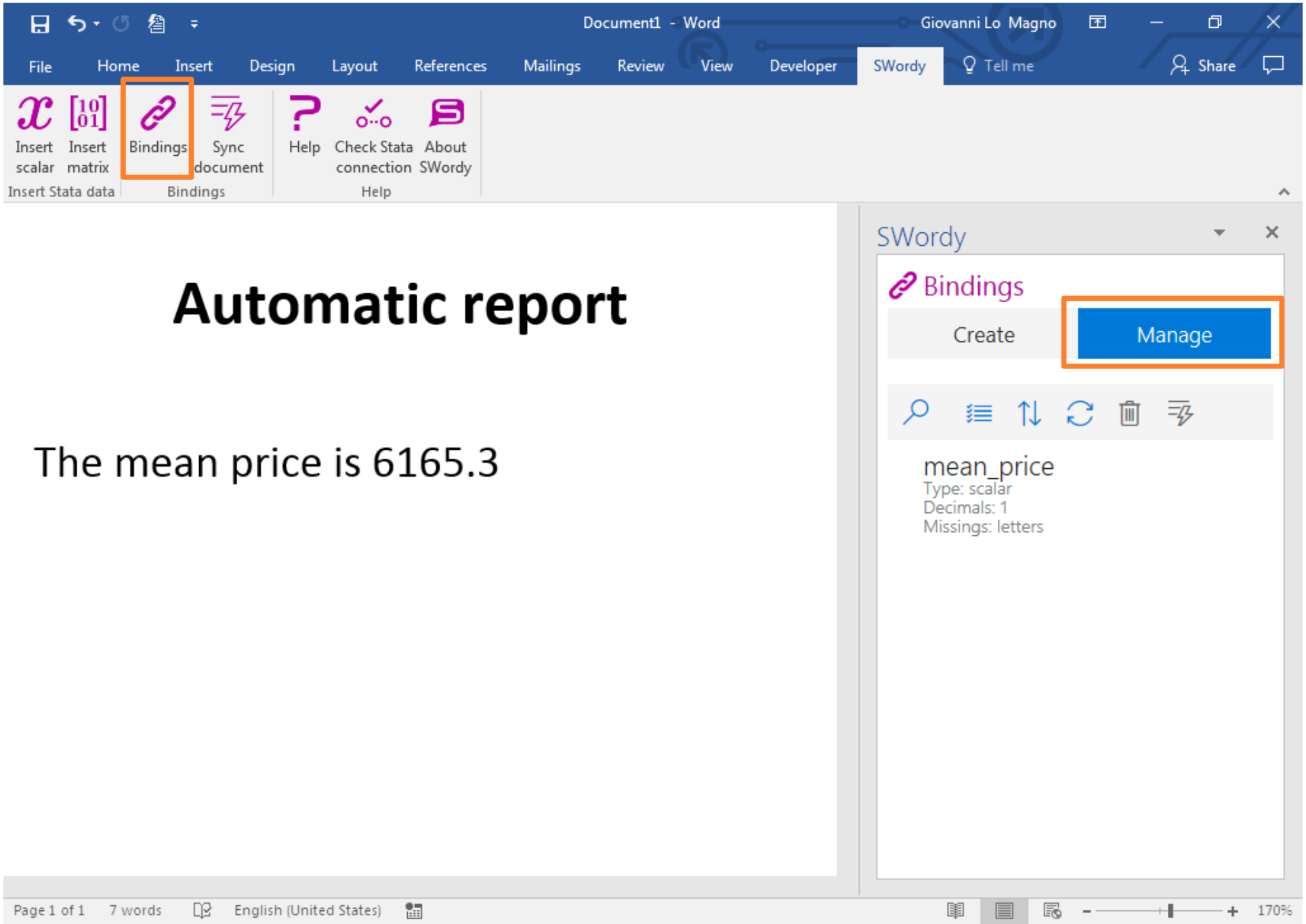
Binding to a Stata scalar – Step 3: set the binding



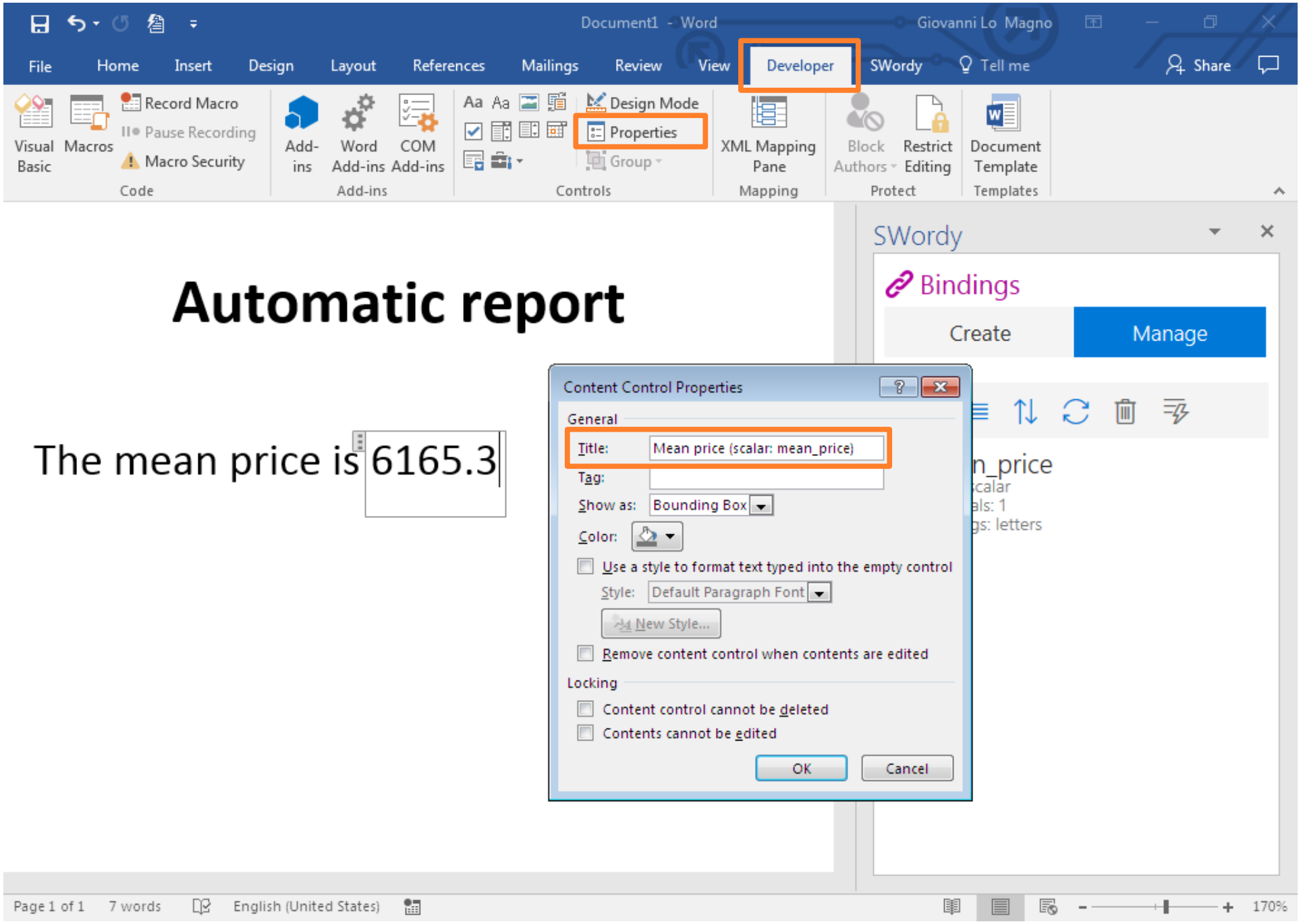
Sync the document



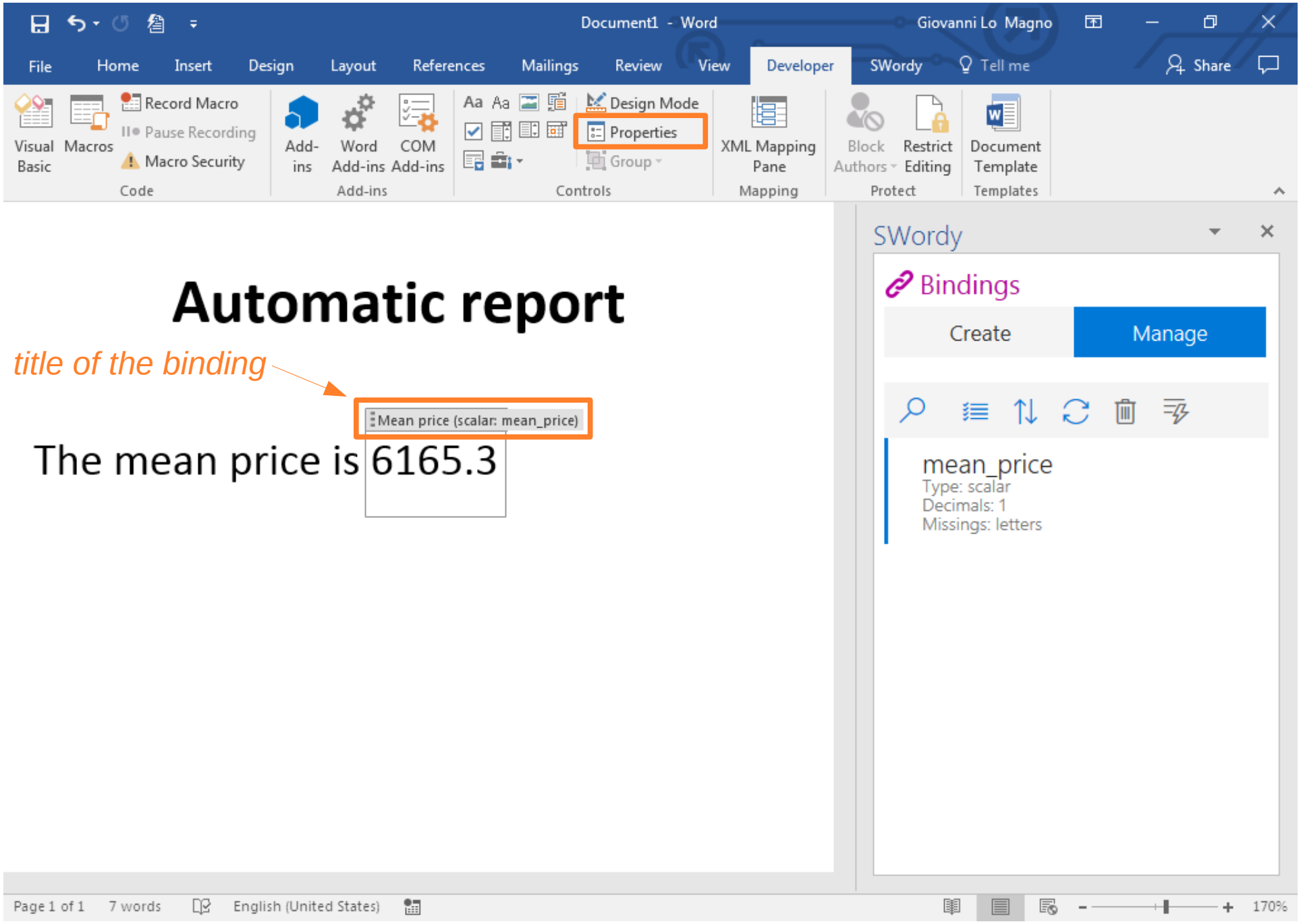
Managing bindings



Adding a title to the binding



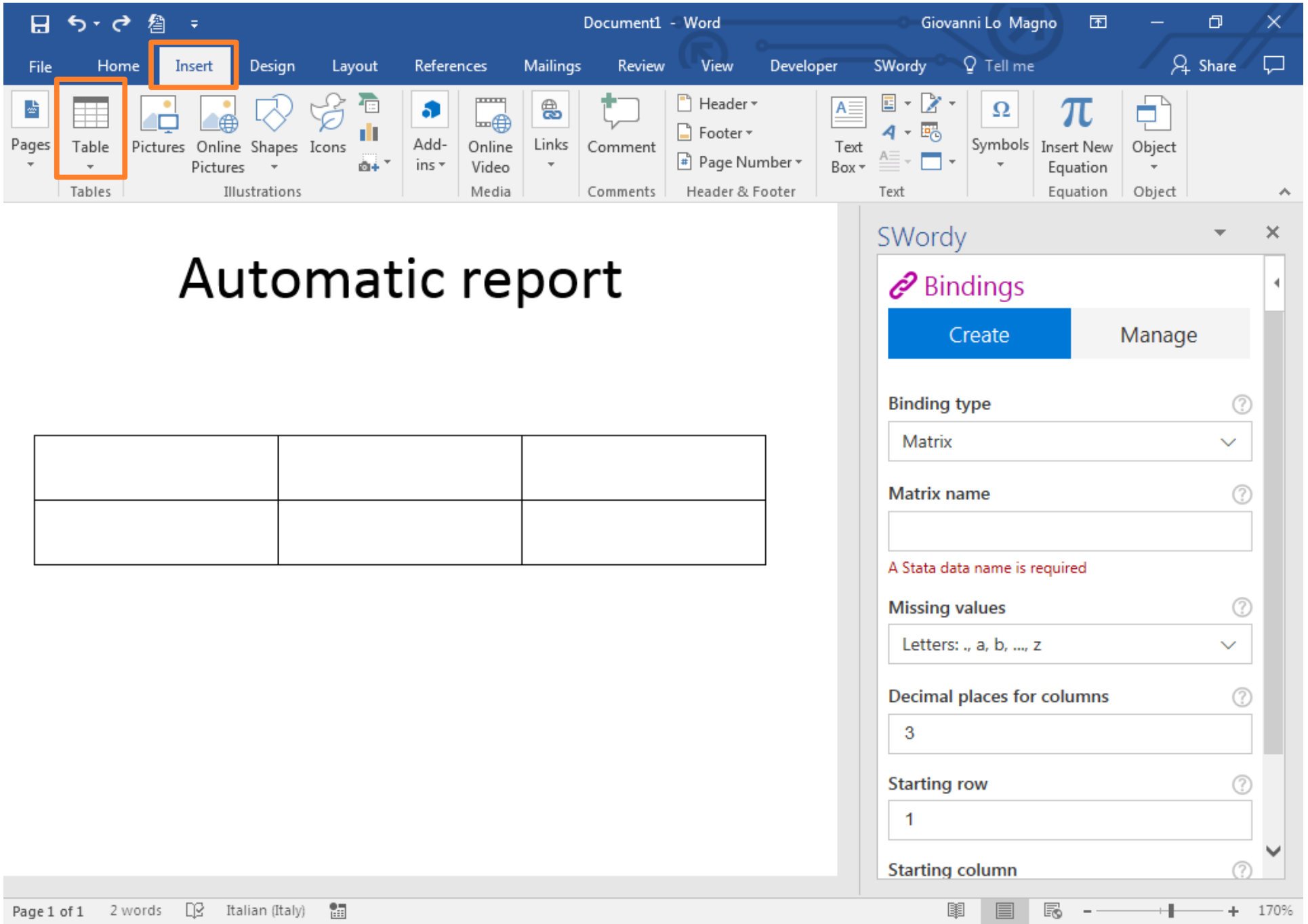
Clicking on the binding to read its title



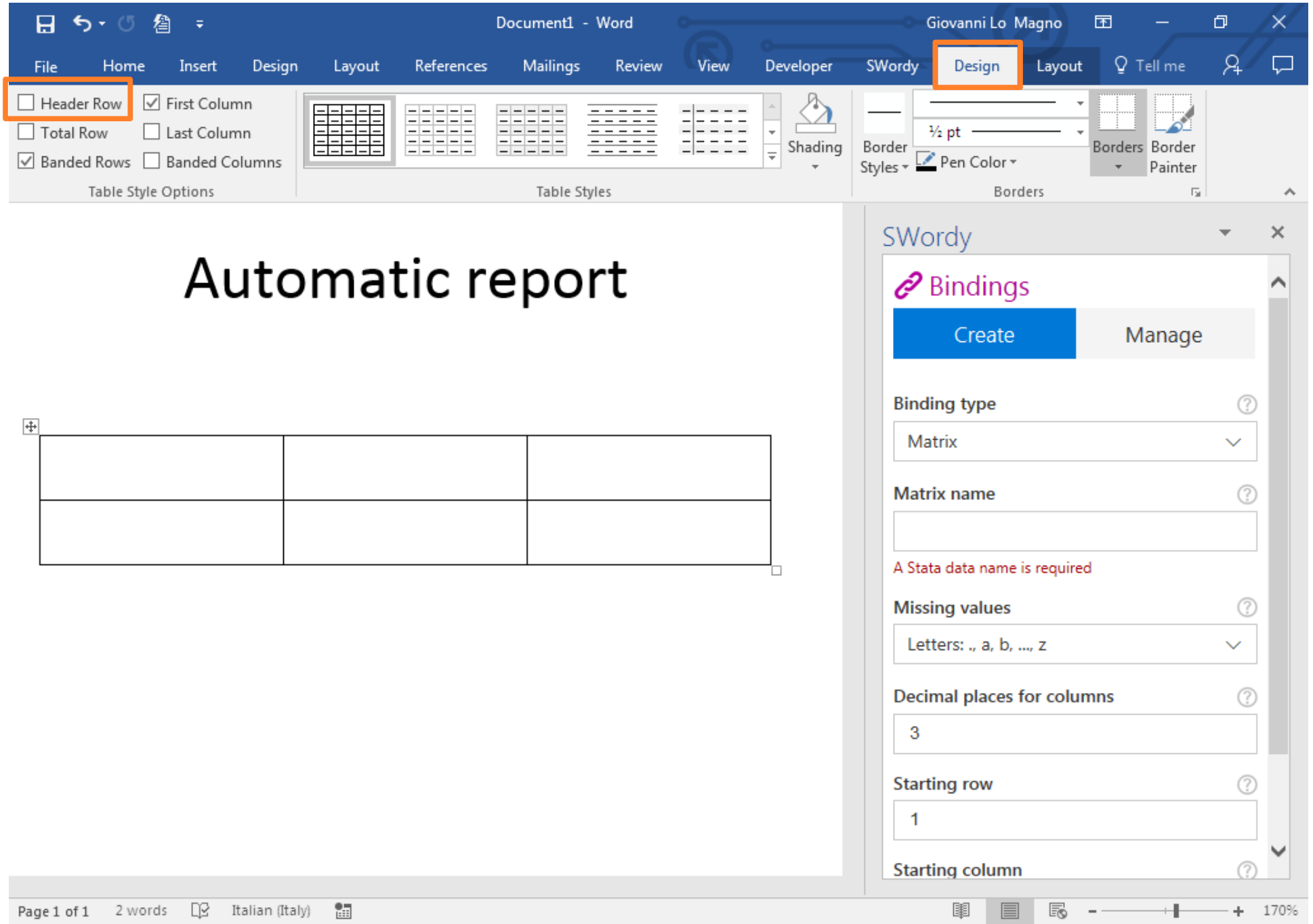
Binding to a Stata matrix – Step 0: optionally prepare the Stata matrix

```
. matrix mymatrix = 11, 12, 13 \ 21, 22, 23

. matrix list mymatrix

mymatrix[2,3]
    c1  c2  c3
r1  11  12  13
r2  21  22  23
```

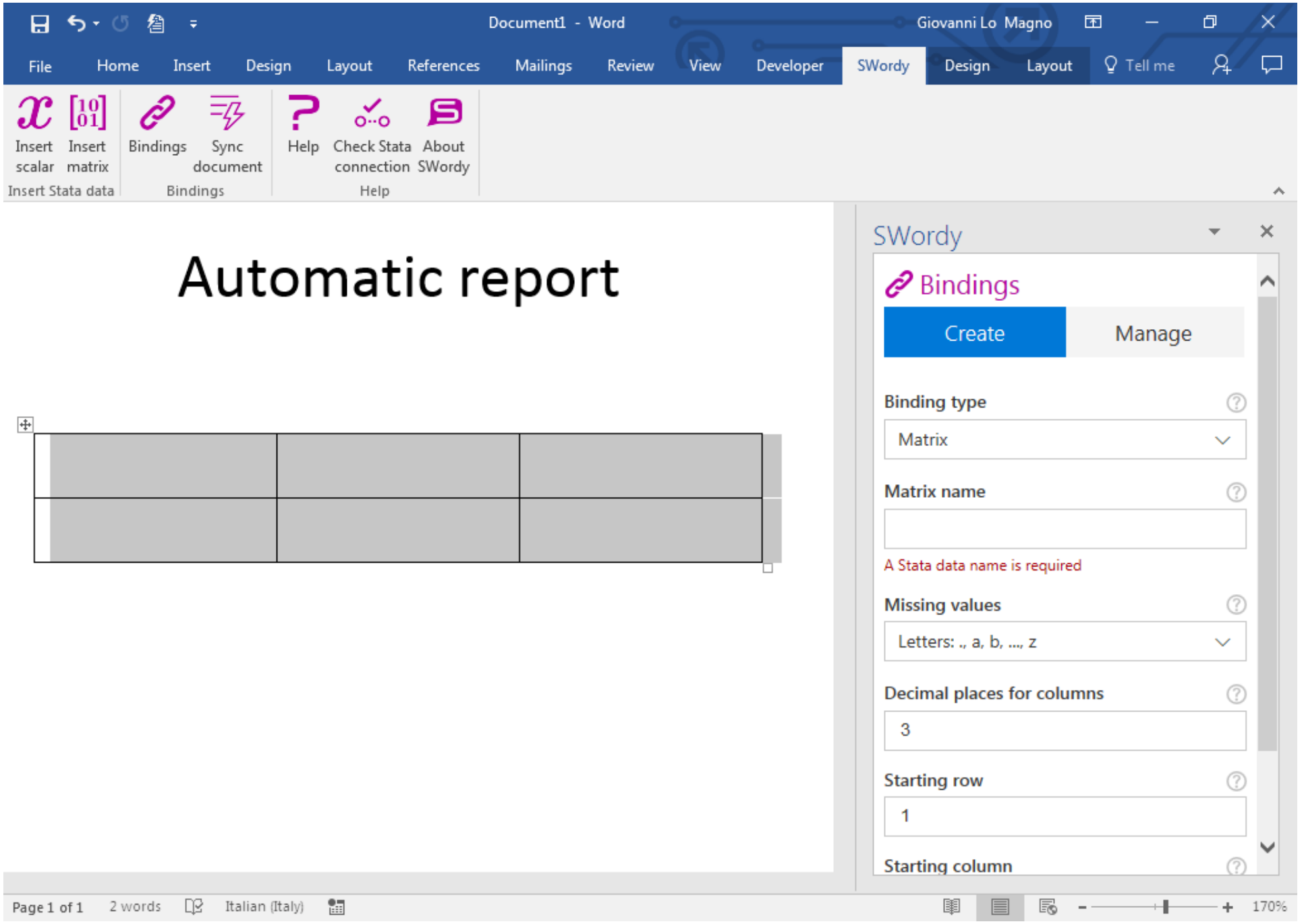
Binding to a Stata matrix – Step 1: insert a table in the document



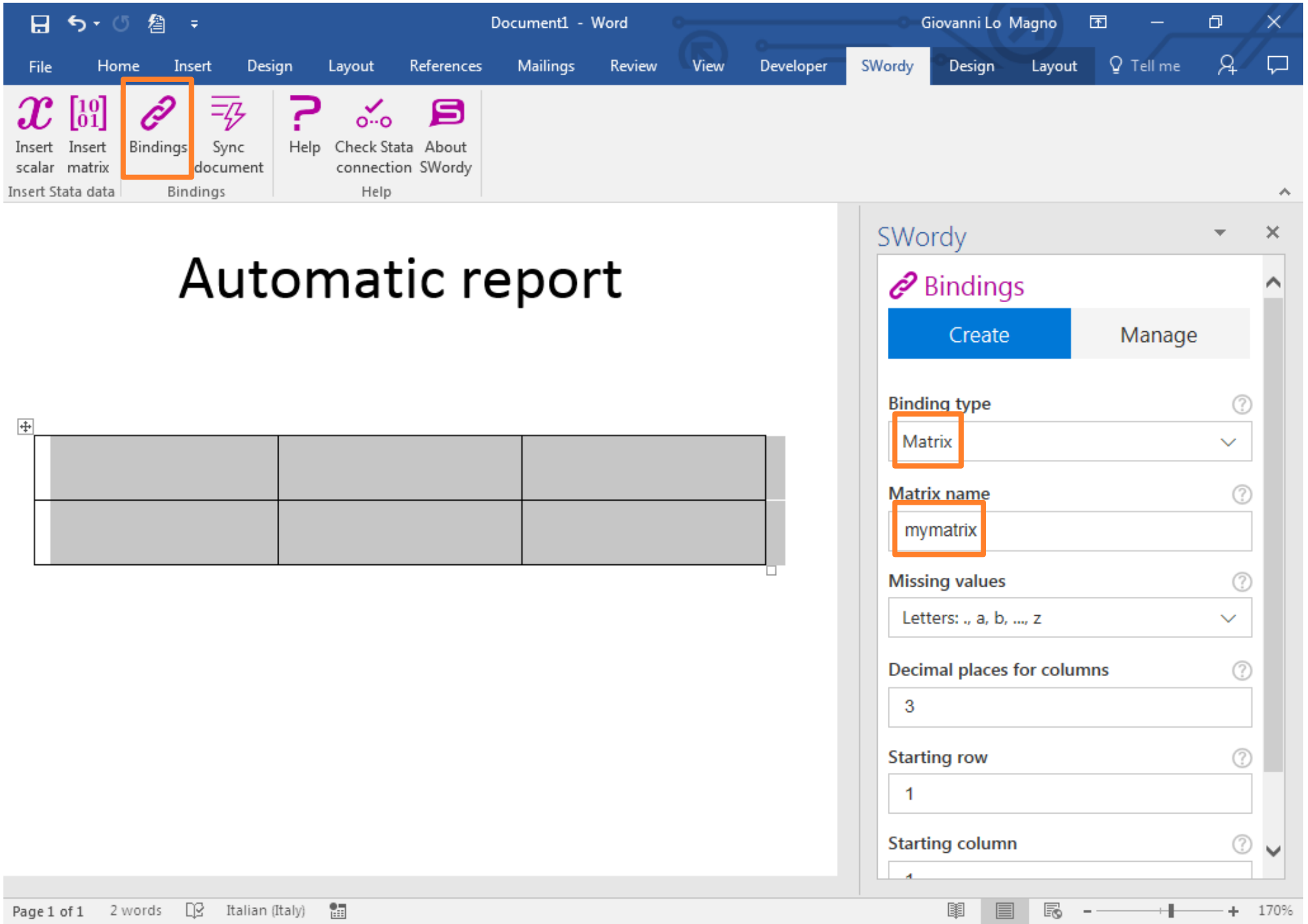
Binding to a Stata matrix – Step 2: disable the table header row



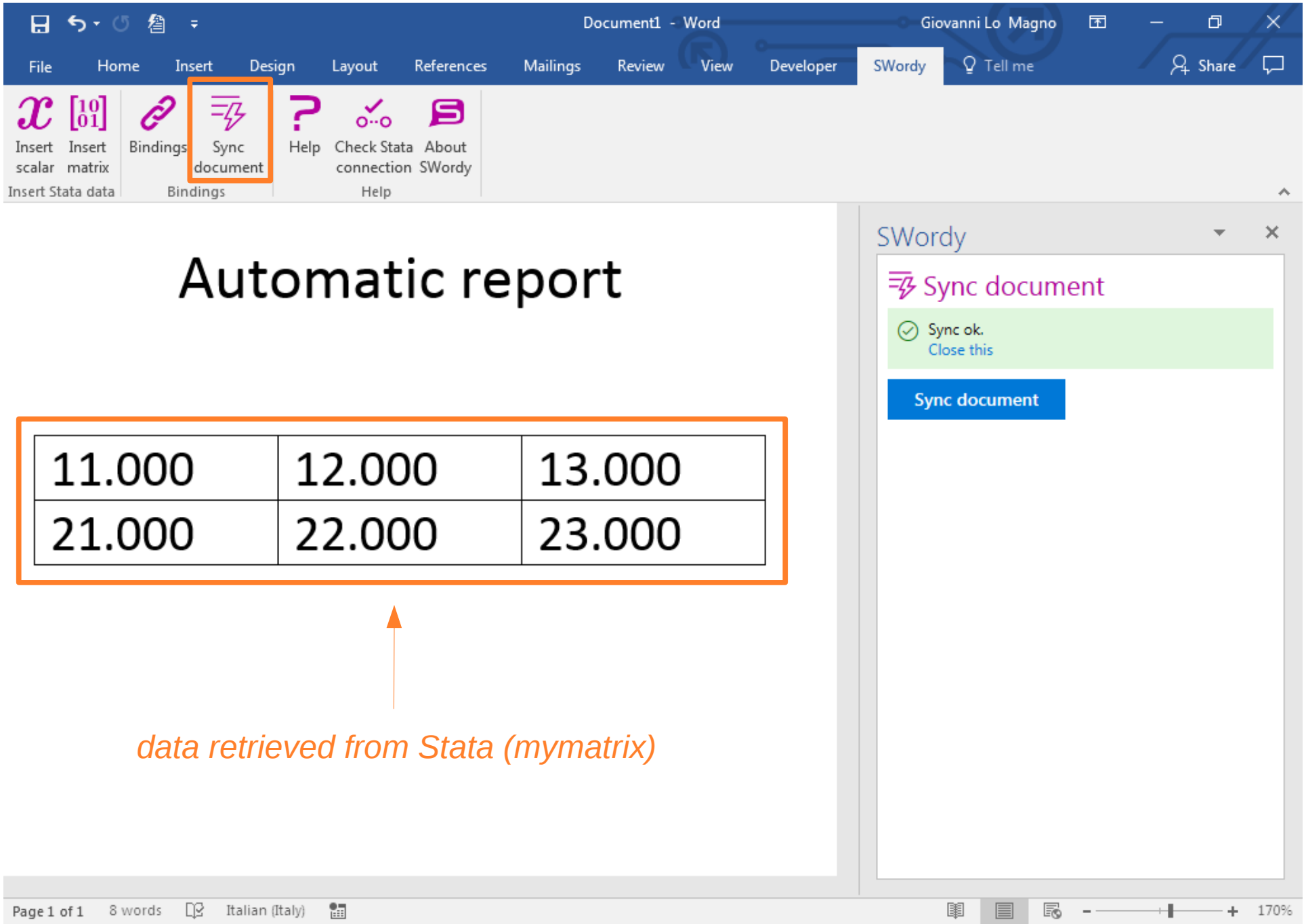
Binding to a Stata matrix – Step 3: select the whole table



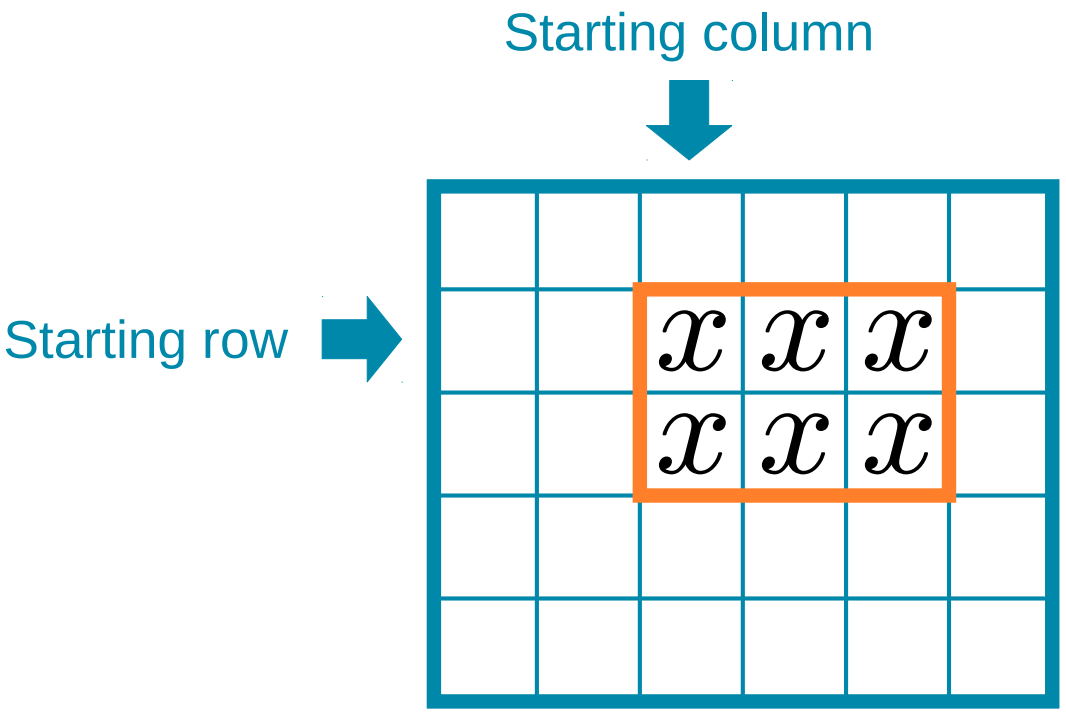
Binding to a Stata matrix – Step 4: set the binding

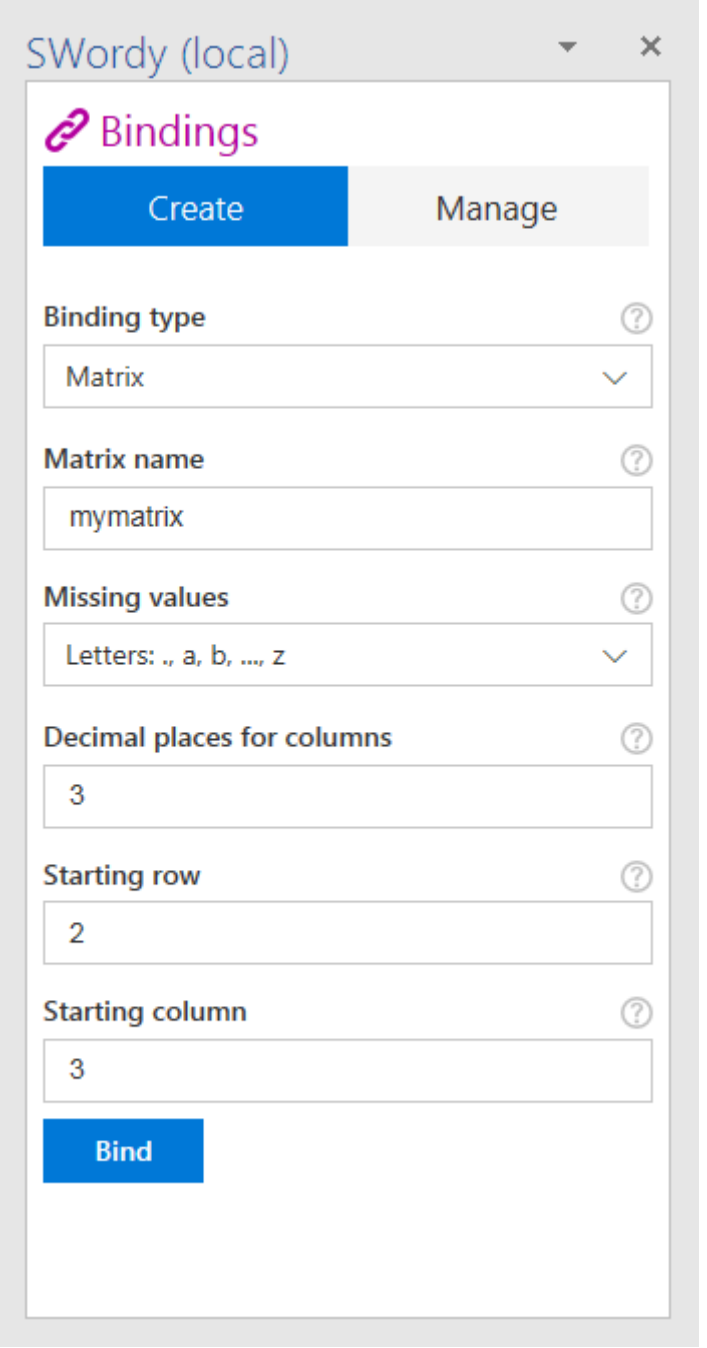


Sync the document



Starting row and starting column for matrices

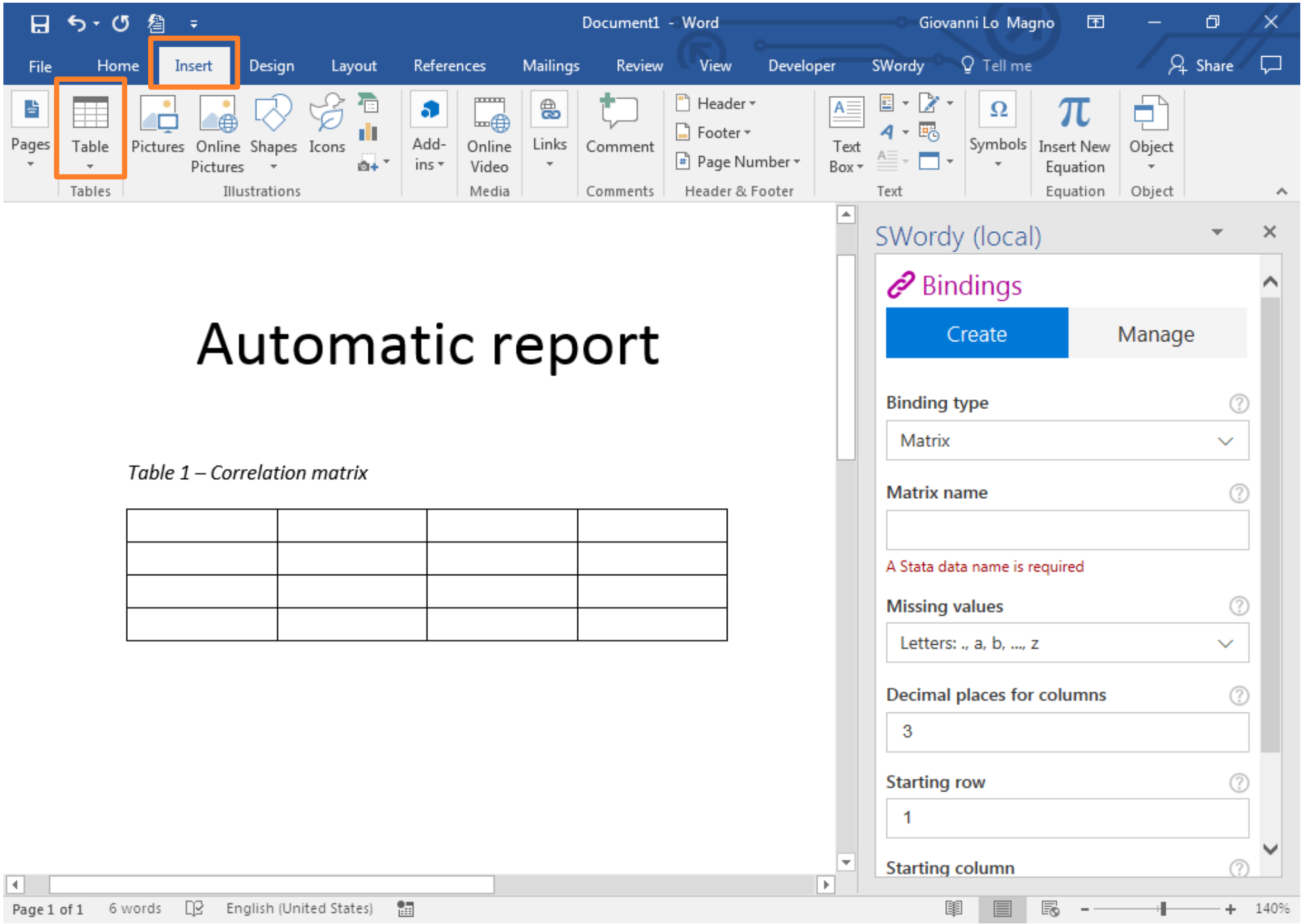




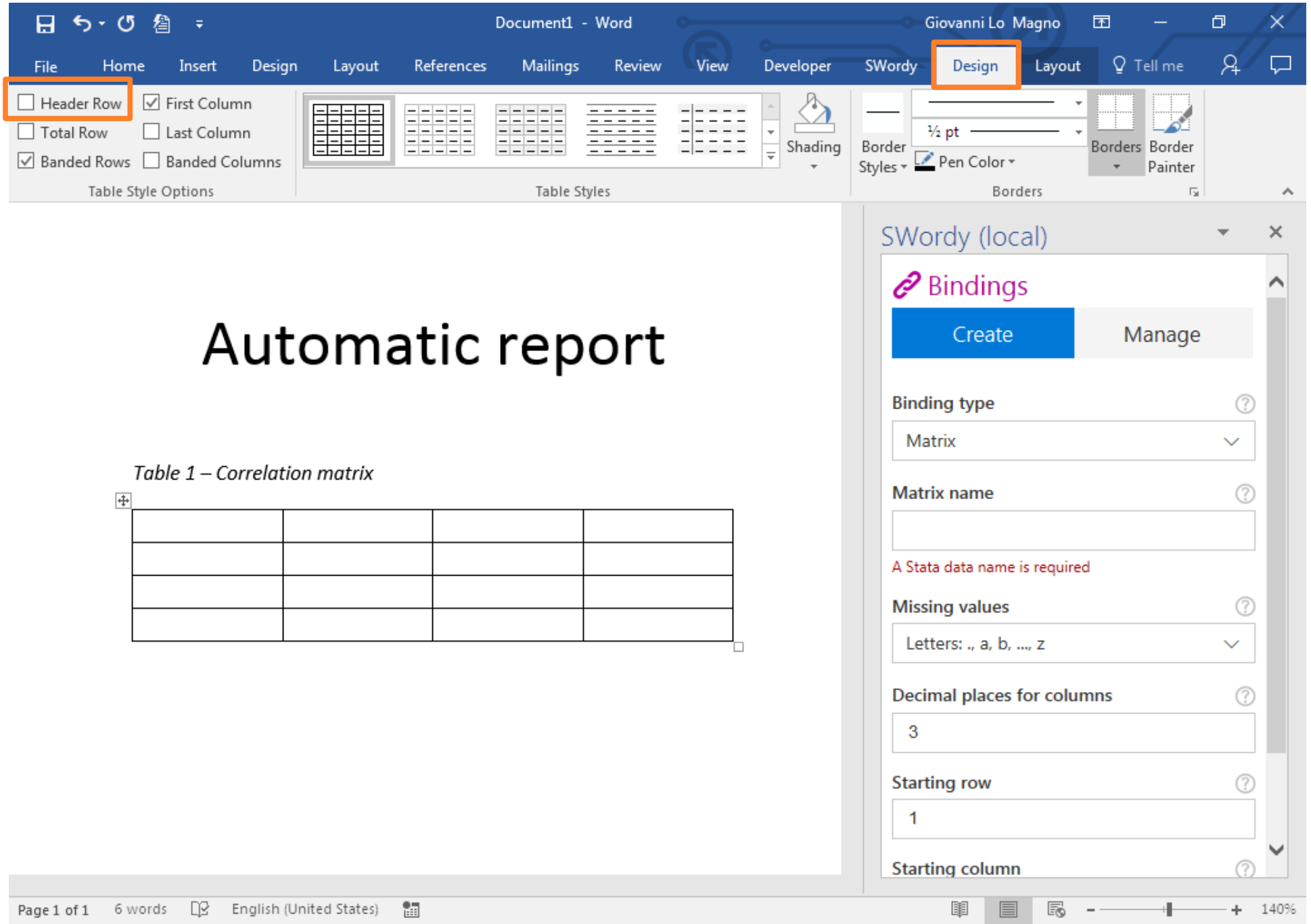
Matrix with pre-existing headers – Step 0: optionally prepare the matrix

```
. sysuse auto
(1978 Automobile Data)

. correlate price weight length
(obs=74)

             |    price   weight   length
-------------+---------------------------
       price |   1.0000
      weight |   0.5386   1.0000
      length |   0.4318   0.9460   1.0000


. matrix corr_matrix = r(C)
```

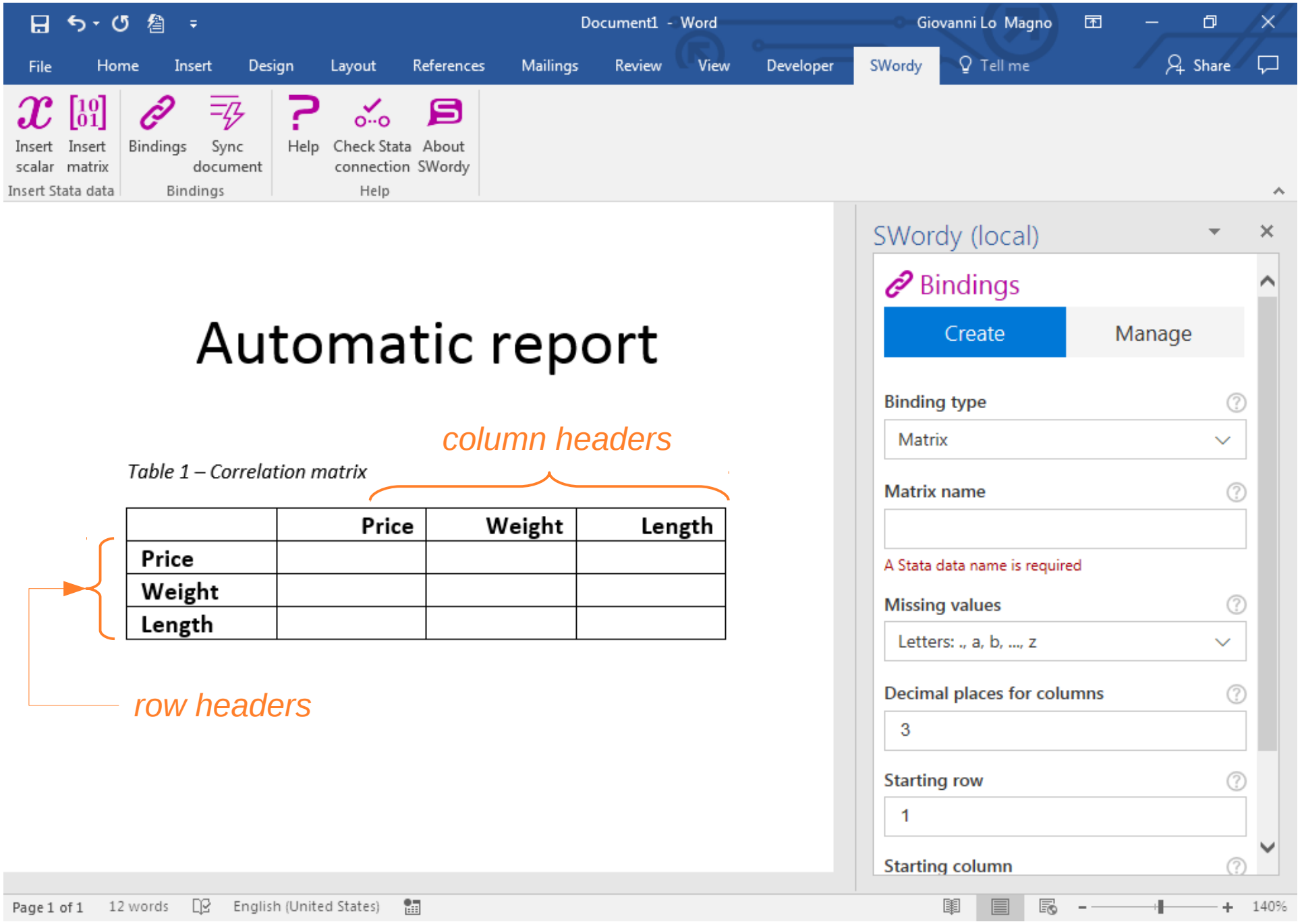
Matrix with pre-existing headers – Step 1: insert the table



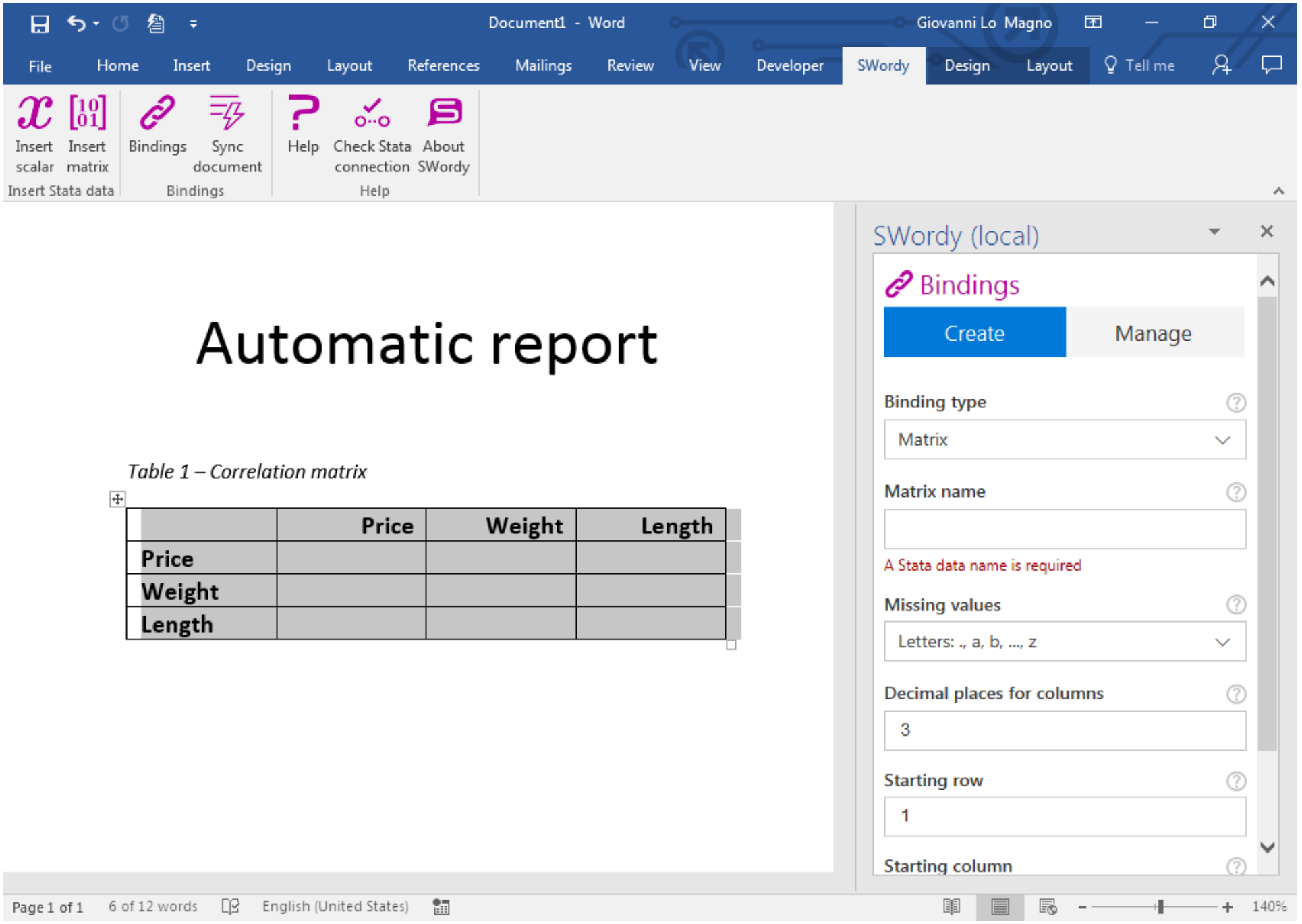
Matrix with pre-existing headers – Step 2: disable header row



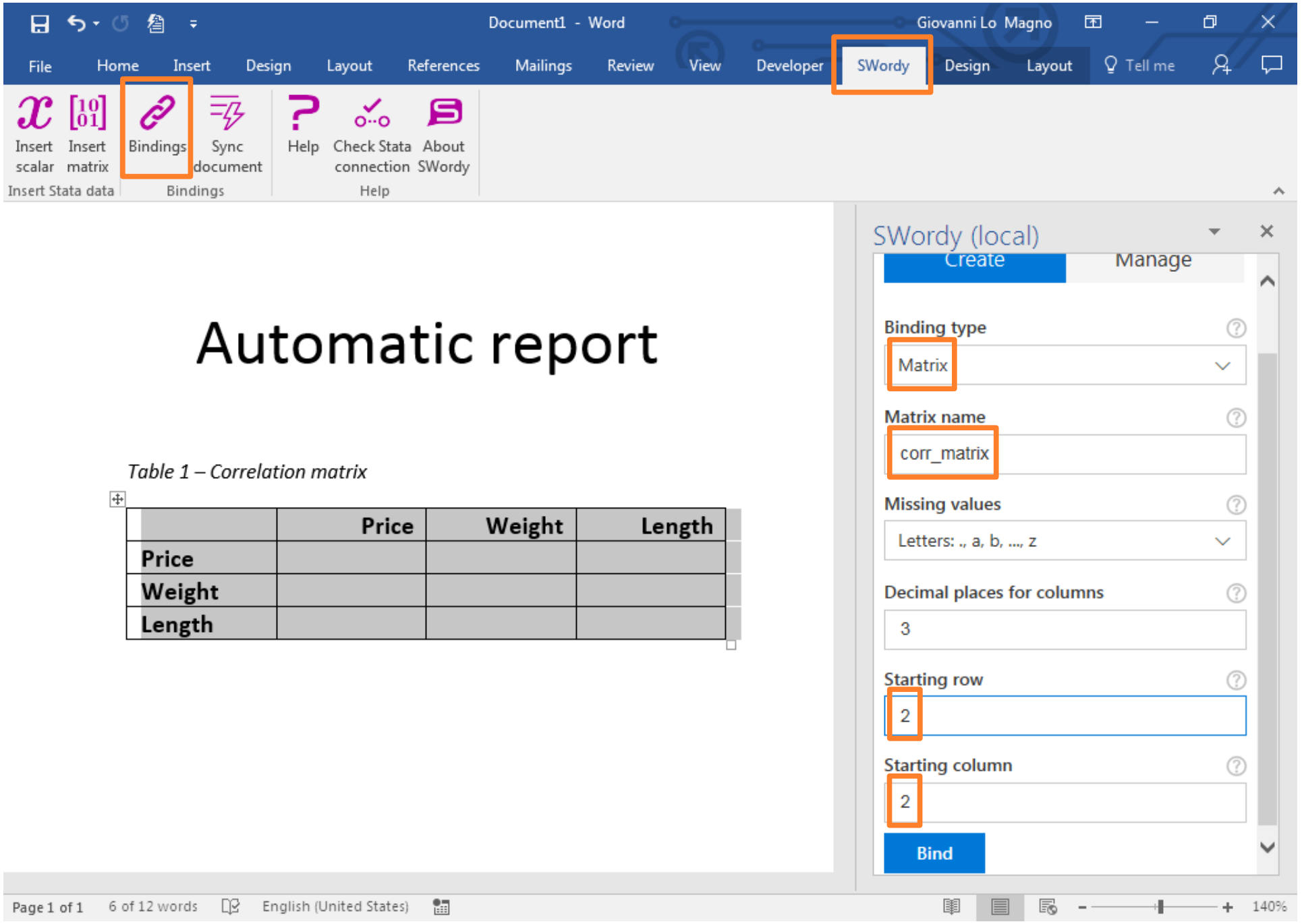
Matrix with pre-existing headers – Step 3: type the headers



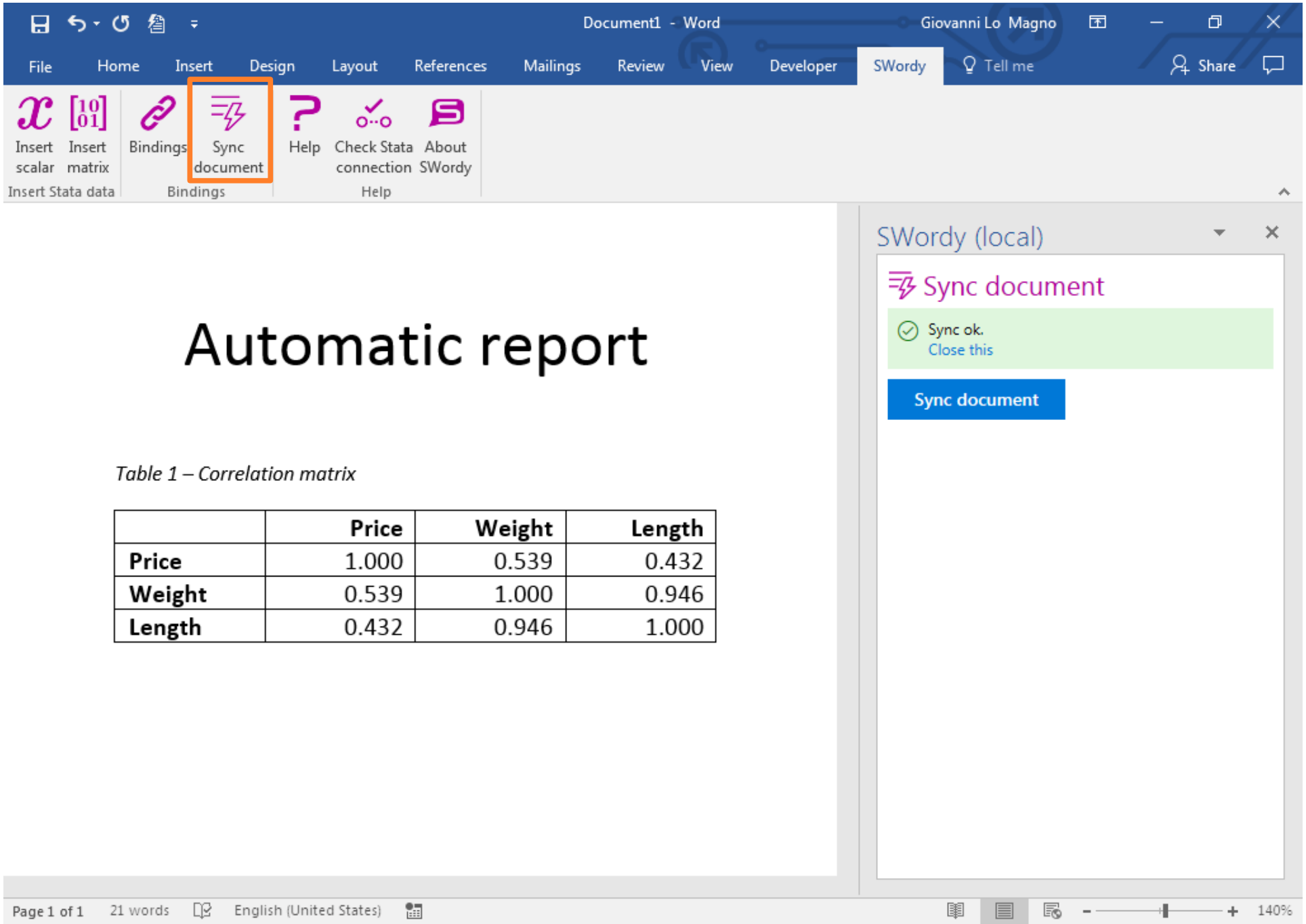
Matrix with pre-existing headers – Step 4: select the whole table



Matrix with pre-existing headers – Step 5: set the binding



Matrix with pre-existing headers – Step 6: sync



	Price	Weight	Length
Price	1.000	0.539	0.432
Weight	0.539	1.000	0.946
Length	0.432	0.946	1.000

Missing values in Stata

They are the largest 27 numbers allowed by the particular storage type

For double-precision storage type:

```
.   = 8.98846567431157 95E+307
.a  = 8.99066012393909 7E+307
.b  = 8.99285457356661 45E+307
 ⋮
.z  = 9.04552136462703 39E+307
```

Order:

. < **.a** < **.b** < … < **.z**

Note: scalars and matrices are stored in double precision

Missing value representation in SWordy

```
. matrix matrix_with_missings = ., .a, .b

. mat list matrix_with_missings

matrix_with_missings[1,3]
    c1  c2  c3
r1   .  .a  .b
```

Letters

.	.a	.b

Letters in parentheses

(.)	(a)	(b)

Dash

-	-	-

Dot

.	.	.

Missing value representation in SWordy

```
. matrix matrix_with_missings = ., .a, .b

. mat list matrix_with_missings

matrix_with_missings[1,3]
    c1  c2  c3
r1   .  .a  .b
```

Letter “m”

m	m	m

Acronym “NA”

NA	NA	NA

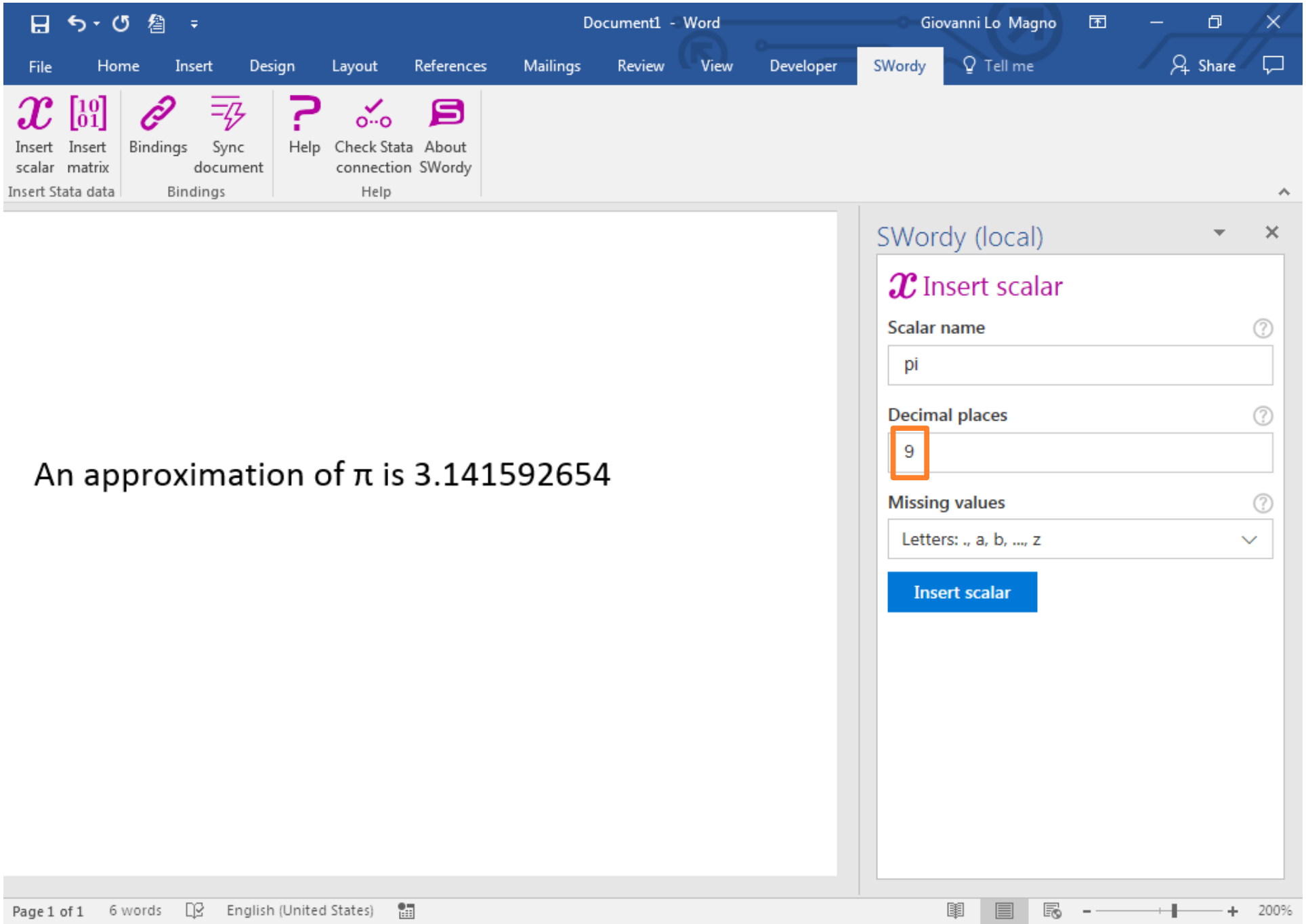
Acronym “NaN”

NaN	NaN	NaN

IEEE 754

8.98846567431158e+307	8.99066012393909 7e+307	8.99285457356661 4e+307

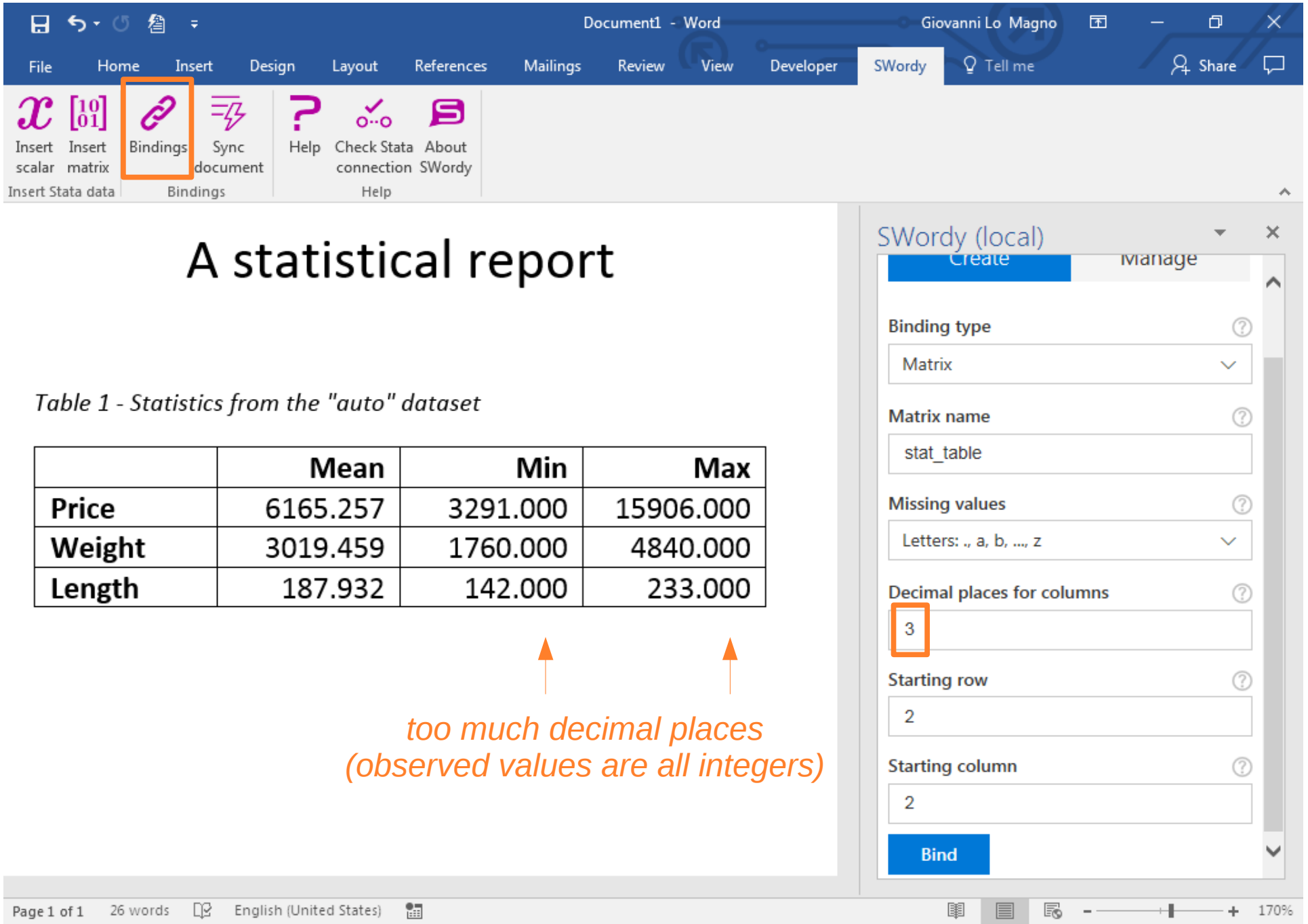
Decimal places for scalars



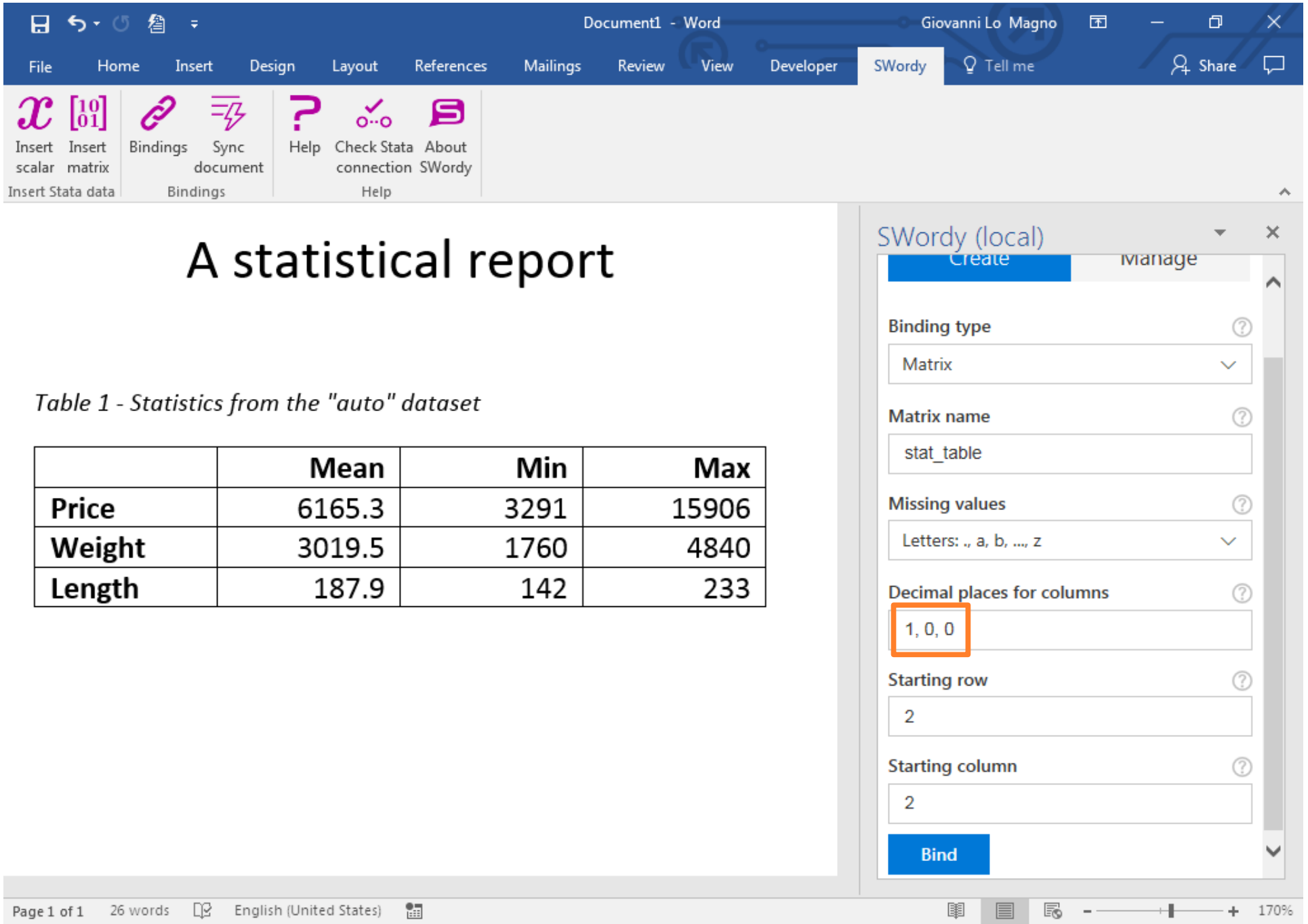
Decimal places for matrices: preparing the example data

```
. sysuse auto
(1978 Automobile Data)

. tabstat price weight length, statistics(mean min max) save

   stats |     price    weight    length
---------+------------------------------
    mean |  6165.257  3019.459  187.9324
     min |      3291      1760       142
     max |     15906      4840       233
-------------------------------------------

. matrix stat_table = r(StatTotal)'

. matrix list stat_table

stat_table[3,3]
              mean         min         max
 price  6165.2568        3291       15906
weight  3019.4595        1760        4840
length  187.93243         142         233
```

Decimal places for matrices: same format for all the columns



Decimal places for matrices: custom format for each column



Specifying decimal places for columns: shorter syntax

Example

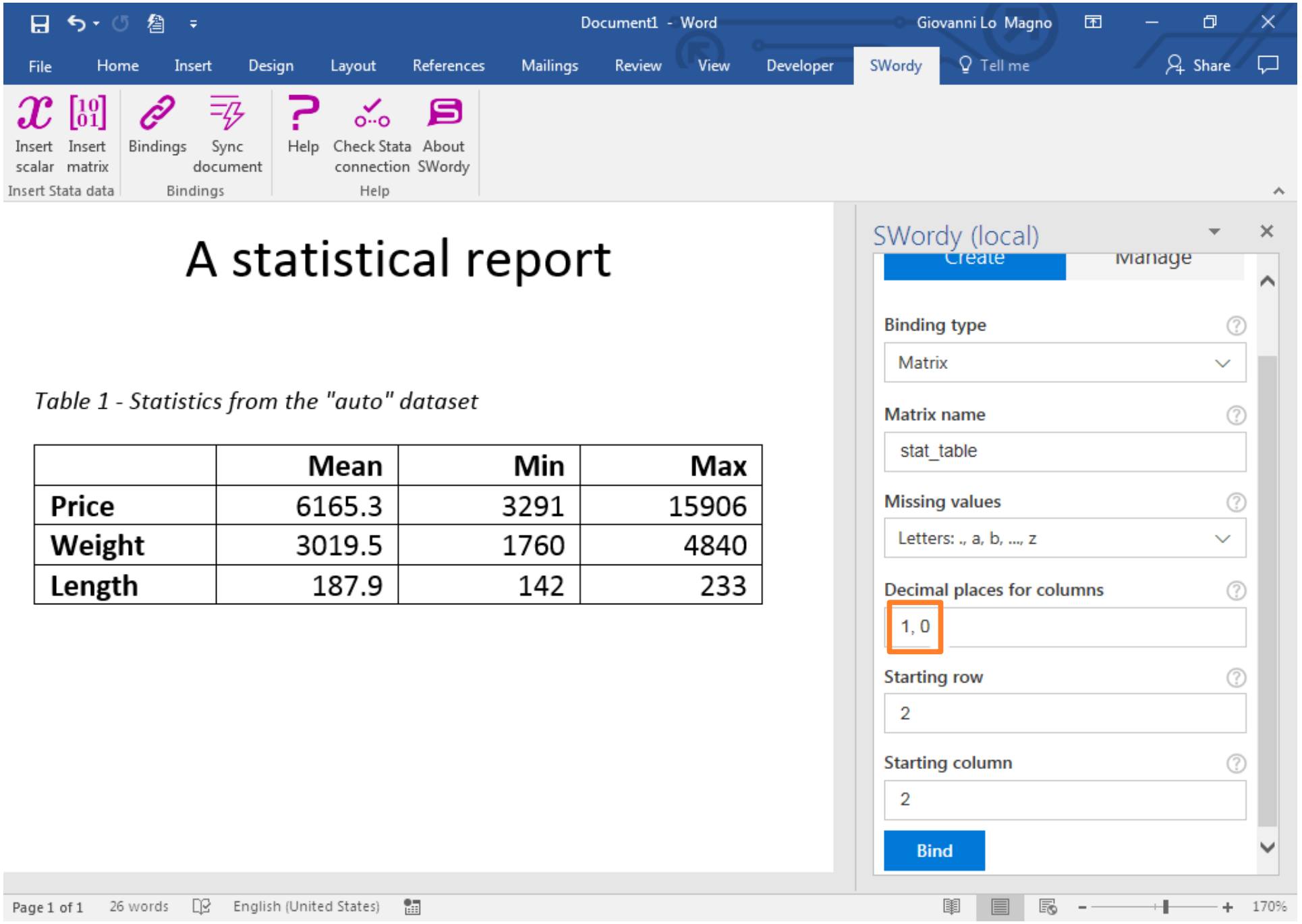
The number of columns is 7, but you specified only two:

6, 3

...this is equivalent to:

6, 3, 3, 3, 3, 3, 3

An example of shorter syntax for specifying decimal places for columns



Usage modes

From the internet server

Pros:

- easy installation
- SWordy will be always updated

Cons:

- SWordy runs slower

Locally

Pros:

- SWordy runs faster

Cons:

- difficult installation
- an SSL Certificate must be trusted

The future?

Thank you for your attention

Appendices

- Appendix A – `swire` command syntax
- Appendix B – The SWire protocol
- Appendix C – SWire special methods
- Appendix D – SWire4js
- Appendix E – SQuery

Appendix A

`swire` command syntax

The `swire` commands suite

- `swire`
- `swire start`
- `swire stop`
- `swire restart`
- `swire status`
- `swire methods`
- `swire initsecurity`
- `swire exportcert`
- `swire certinfo`
- `swire version`

Syntax of `swire`

Syntax

`swire`

Description

Get the list of the SWire commands.

Examples

```
. swire
SWire commands:

swire start         Start the SWire HTTP server
swire start, https Start the SWire HTTPS server
swire stop          Stop the SWire server
swire restart       Restart the SWire server
swire status        Display the status of the SWire server
swire methods       Display the exposed Stata-Java API methods and the SWire special methods
swire initsecurity Generate the SWire keystore
swire exportcert    Export the SWire Certification Authority Certificate from the SWire keystore
swire certinfo      Display info about the certificates
swire version       Display the SWire version number
```

Syntax of `swire start`

Syntax

`swire start [, http|https port(`*`port_number`*`)]`

Description

Start the SWire server.

Options

`http|https` the `http` option specifies the HTTP mode and the `https` option specifies the HTTPS mode. `http` is the default.

`port_number` is the port number to which the SWire server will attempt to bind. The default value is 50000. An integer number between 1024 and 65535 must be specified.

Examples

```
. swire start, https port(8088)
-------------------------------------------------
The SWire HTTPS server is listening on port 8088
-------------------------------------------------
(output omitted)
```

Syntax of `swire stop`

Syntax

```
swire stop
```

Description

Stop the SWire server.

Examples

```
. swire stop
The SWire HTTP server has stopped listening on port 50000
```

Syntax of `swire restart`

Syntax

`swire restart, http|https port(`*`port_number`*`)`

Description

Stop and start the SWire server.

Options

`http|https` the `http` option specifies the HTTP mode and the `https` option specifies the HTTPS mode. `http` is the default. If neither has been specified and the SWire server is listening then the server will be restarted in the current mode.

`port_number` is the port number to which the SWire server will try to bind. The default value is 50000. An integer number between 1024 and 65535 must be specified.

Examples

```
. swire restart, https
-------------------------------------------
SWire HTTPS server is listening on port 5000
-------------------------------------------
(output omitted)
```

Syntax of `swire status`

Syntax

```
swire status
```

Description

Display the status of the SWire server.

Examples

```
. swire status
The SWire HTTP server is listening on port 50000
```

Syntax of `swire methods`

Syntax

```
swire methods
```

Description

Display the Stata-Java API methods and the SWire special methods which have been exposed by SWire. A corresponding availability status will also be reported for each Stata-Java API method. A Stata-Java API method is unavailable when the method is not available in the Stata-Java API, which is used by your Stata release.

Examples

```
. swire methods
-------------------------------------------------------------------------------
Stata-Java API methods
-------------------------------------------------------------------------------
com.stata.sfi.Data.addVarByte(class java.lang.String) - available: yes
com.stata.sfi.Data.addVarDouble(class java.lang.String) - available: yes
com.stata.sfi.Data.addVarFloat(class java.lang.String) - available: yes

(output omitted)
```

Syntax of `swire initsecurity`

Syntax

```
swire initsecurity [, replace]
```

Description

Generate the SWire keystore.

Options

`replace` forces the command to replace the existing SWire keystore.

Examples

```
. swire initsecurity
(output omitted)
```

Syntax of `swire exportcert`

Syntax

`swire exportcert [using` *`filename`*`] [, replace]`

Description

Export the SWire Certification Authority Certificate from the SWire keystore to an output file.

Options

`using` specifies the output file.

`replace` forces the command to replace an already existing output file.

Examples

```
. swire exportcert using "swire_ca.crt"
The SWire Certification Authority Certificate was exported from the SWire keystore file

from: "/home/giovanni/swire.keystore"
to: "/home/giovanni/swire_ca.crt"
```

Syntax of `swire certinfo`

Syntax

```
swire certinfo
```

Description

Display info about the certificates in the SWire keystore.

Examples

```
. swire certinfo
(output omitted)
```

Syntax of `swire version`

Syntax

```
swire version
```

Description

Display the SWire version number.

Examples

```
. swire version
SWire 0.2
```

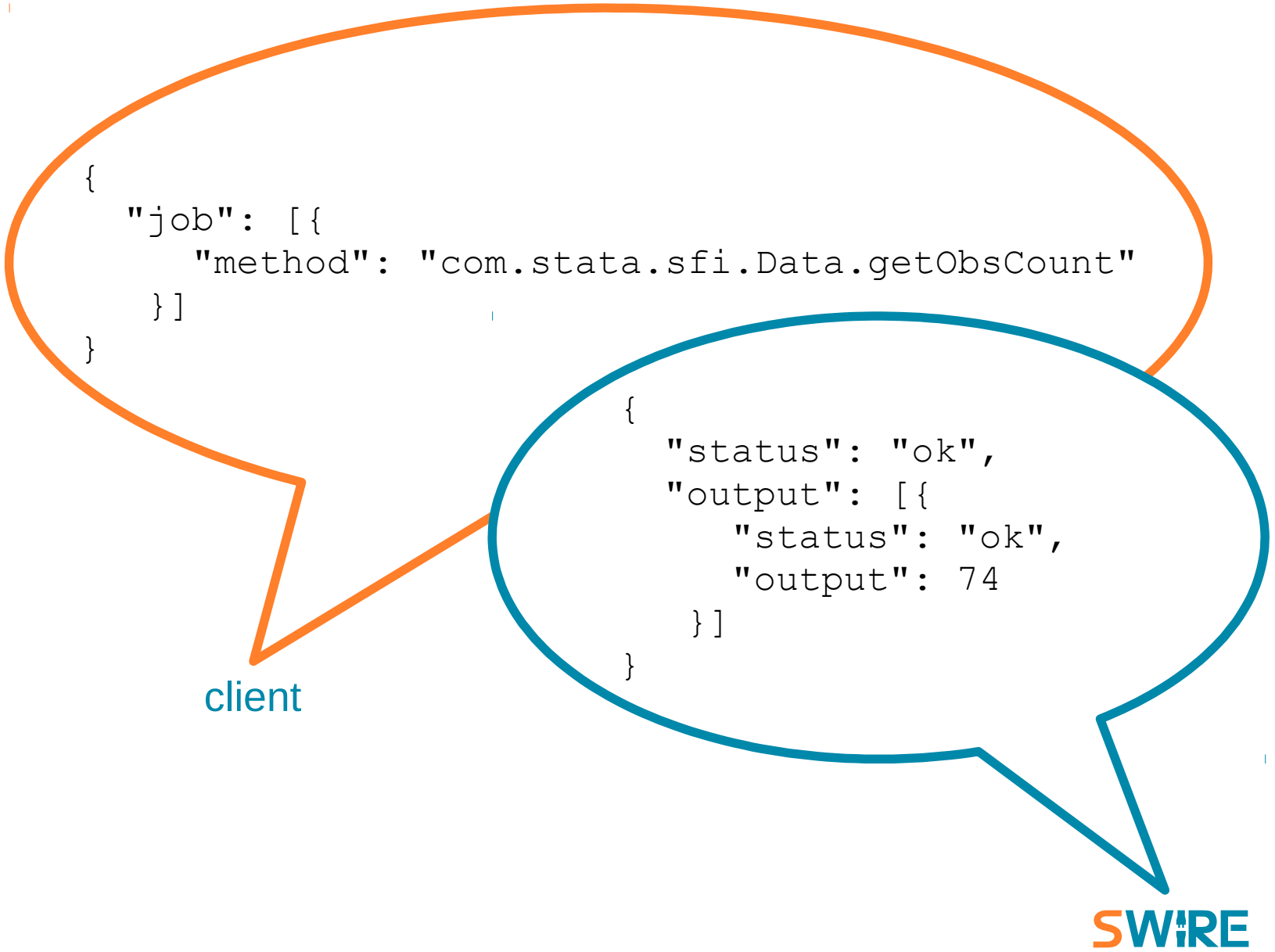
Appendix B

The SWire protocol

The basics of the SWire protocol

- based on the HTTP protocol
- the body of the HTTP request/response is a SWire request/response
- the HTTP body is a base64 encoding of a MessagePack serialization of a JSON request/response
- the syntax can be described by JSON strings
- the official MessagePack website: http://msgpack.org/

An example: asking for the number of observations



Serialization of a SWire JSON request/response

JSON plain text
(human readable)

```
{"job":[{"method":"com.stata.sfi.Data.getObsCount"}]}
```

Binary MessagePack serialization
(high compression and no numeric precision loss)

```
81 a3 6a 6f 62 91 81 a6 6d 65 74 68 6f 64 be 63 6f 6d 2e
73 74 61 74 61 2e 73 66 69 2e 44 61 74 61 2e 67 65 74 4f
62 73 43 6f 75 6e 74
```

Base64 encoding
(required for the HTTP body)

```
gaNqb2KRgaZtZXRob2S+Y29tLnN0YXRhLnNmaS5EYXRhLmdldE9ic0NvdW50
```

The final HTTP request

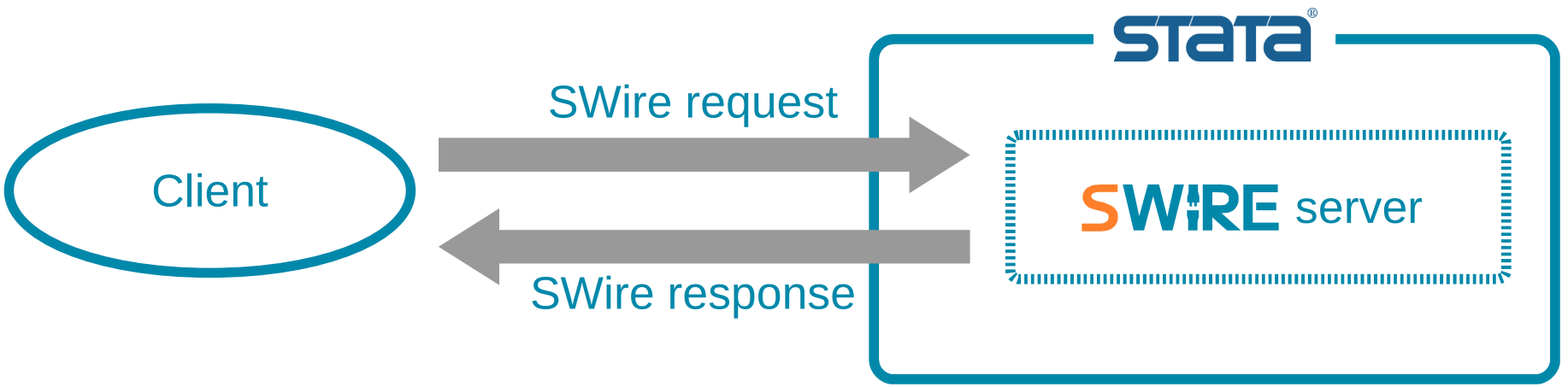
```
POST / HTTP/1.1
Host: 127.0.0.1:50000
User-Agent: Mozilla/5.0 (X11; Ubuntu; Linux i686; rv:47.0)
Gecko/20100101 Firefox/47.0
Accept: */*
Accept-Language: en-US,en;q=0.5
Accept-Encoding: gzip, deflate
Content-Type: application/x-www-form-urlencoded; charset=UTF-8
Content-Length: 60
Origin: null
Connection: keep-alive
Cache-Control: max-age=0

gaNqb2KRgaZtZXRob2S+Y29tLnN0YXRhLnNmaS5EYXRhLmdldE9ic0NvdW50
```

This is the body

SWire messages

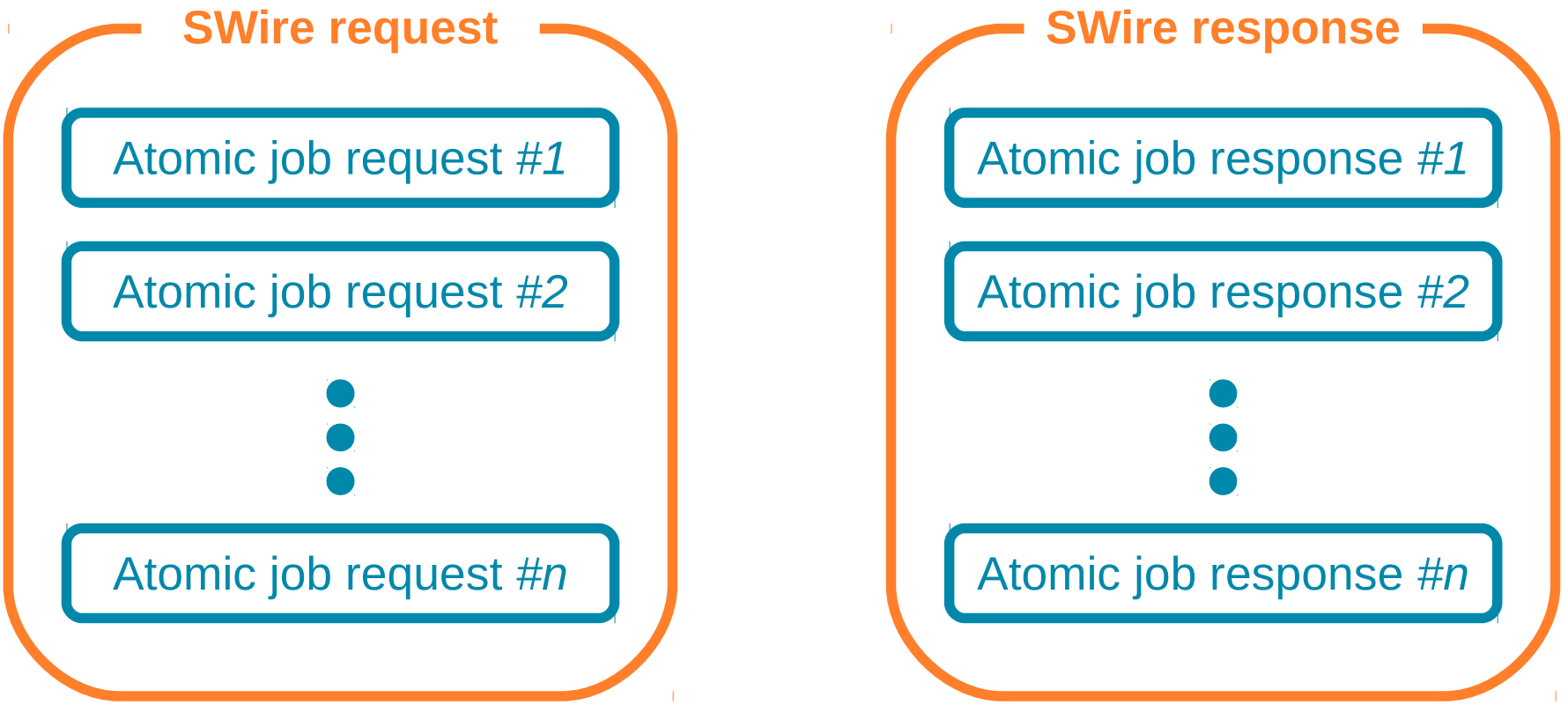
Two types of SWire messages:

- SWire request
- SWire response



SWire request/response and its atomic job request/response

- A SWire request consists of atomic job requests
- A SWire response consists of atomic job responses
- Each atomic job request is a call to a Stata-Java method or a SWire special method
- Each atomic job response is the output corresponsing to an atomic job request



The SWire request: syntax and description

Syntax

```
{
  "continue": continueFlag,
  "job": jobArray
}
```

where:

`continueFlag`

Type: boolean
Required: no
Default value: false

`jobArray`

Type: array
Required: yes

Description

`continueFlag`

Controls if the execution of the array of atomic jobs must halt when an atomic job reports an error.

`jobArray`

An array of atomic jobs (more about atomic jobs later).

The SWire request: an example

```
{
  "continue": true,
  "job": [
    {
      "method": "com.stata.sfi.Scalar.setValue",
      "args": ["myscalar", 21.3]
    },
    {
      "method": "com.stata.sfi.Data.addVarDouble",
      "args": ["myvar"]
    }
  ]
}
```

- two Stata Java API methods are invoked
- the “com.stata.sfi.Scalar.setValue” method is invoked with the arguments “myscalar” and 21.3
- the “com.stata.sfi.Data.addVarDouble” method is invoked with only the “myvar” argument
- the “continue” flag is set to true, thus the second atomic job will be executed even if the first atomic job reports an error

The atomic job: syntax and description

Syntax

```
{
  "method": method,
  "args": args
}
```

where:

`method`

Type: string
Required: yes

`args`

Type: array or map
Required: it depends on the method

`method`

It can be either a Stata-Java Api method (e.g.: “com.stata.sfi.Scalar.setValue”) or a special method (e.g.: “$appendRow”). Note that all special methods begin with “$”.

`args`

It is a parameters array in the case of a Stata-Java Api method and a map in the case of a special method (more later about the content of `args`).

The atomic job: an example of Stata-Java API call with arguments

Documentation from http://www.stata.com/java/api/:

Package:
`com.stata.sfi`

Class:
`Scalar`

Method:

```
public static int setValue(String name, double val)
```

Parameters:

`name` – scalar name

`val` – value to store in the scalar

Returns:

Return code from Stata; 0 if successful.

The atomic job:

```
{
  "method": "com.stata.sfi.Scalar.setValue",
  "args": ["myscalar", 21.3]
}
```

The atomic job: an example of Stata-Java API call without arguments

Documentation from http://www.stata.com/java/api/:

Package:
`com.stata.sfi`

Class:
`Data`

Method:
```
public static int getVarCount()
```

Returns:
Return the number of variables.

The atomic job:

```
{
  "method": "com.stata.sfi.Data.getVarCount"
}
```

The atomic job: an example of SWire special method call

The atomic job:

```
{
  "method": "$appendRow",
  "args": {
    "data": {
      "myvar1": 8.3,
      "myvar2": 15.7,
      "myvar3": 9.1,
    }
  }
}
```

`args` in the case of a special method call:

- it is a map
- the content depends on the special method (see related documentation)

A complete example of a SWire request with three atomic jobs

atomic job

atomic job

atomic job

```
{
  "job": [
    {
      "method": "com.stata.sfi.Scalar.setValue",
      "args": ["myscalar", 21.3]
    },
    {
      "method": "com.stata.sfi.Data.getVarCount"
    },
    {
      "method": "$appendRow",
      "args": {
        "data": {
          "myvar1": 8.3,
          "myvar2": 15.7,
          "myvar3": 9.1,
        }
      }
    }
  ]
}
```

The continue flag: an example with continue=false

Request

intentionally wrong

```
{
  "continue": false,
  "job": [
    {
      "method": "com.stata.sfi.Data.notExistingMethod"
    },
    {
      "method": "com.stata.sfi.Data.getVarCount"
    }
  ]
}
```

Response

```
    {
      "status": "error_in_job",
      "output": [
        {
          "status": "error",
          "errorType": "STATA_METHOD_NOT_FOUND"
        },
        {
          "status": "not_executed"
        }
      ]
    }
```

not executed because continue=false

The continue flag: an example with continue=true

Request

intentionally wrong

```
{
  "continue": true,
  "job": [
    {
      "method": "com.stata.sfi.Data.notExistingMethod"
    },
    {
      "method": "com.stata.sfi.Data.getVarCount"
    }
  ]
}
```

Response

executed because continue=true

```
    {
      "status": "error_in_job",
      "output": [
        {
          "status": "error",
          "errorType": "STATA_METHOD_NOT_FOUND"
        },
        {
          "status": "ok",
          "output": 12
        }
      ]
    }
```

SWire response

Syntax

```
{
  "status": status,
  "output": output,
  "errorType": errorType
}
```

where:

`status`

A string that can be “ok”, “error” or “error_in_job”. It is “ok” when all the atomic jobs have been correctly executed; it is “error” when there is a syntax error in the request; and it is “error_in_job” when there is an error in one of the atomic jobs. The “status” key always appears in the response.

`output`

An array of atomic job output for each atomic job request. It appears in the response only if “status” is “ok”.

`errorType`

A string that describes the error in the global request. It appears in the response only if “status” is not “ok”.

SWire response depending on the status

Status is “ok”

```
{
  "status": "ok",
  "output": output
}
```

Status is “error”

```
{
  "status": "error",
  "errorType": errorType
}
```

Status is “error_in_job”

```
{
  "status": "error_in_job",
  "errorType": errorType
}
```

SWire response: an example with “ok” status

Request

atomic job request 1

atomic job request 2

```
{
  "job": [
    {
      "method": "com.stata.sfi.Data.getVarCount"
    },
    {
      "method": "com.stata.sfi.SFIToolkit.isValidVariableName",
      "args": ["myvarname"]
    }
  ]
}
```

Response

atomic job response 1

atomic job response 2

```
{
  "status": "ok",
  "output": [
    {
      "status": "ok",
      "output": 12
    },
    {
      "status": "ok",
      "output": true
    }
  ]
}
```

SWire response: an example with “error” status

Request

intentionally wrong

```
{
  "notExistingKey": [
    {
      "method": "com.stata.sfi.Data.getVarCount"
    }
  ]
}
```

Response

```
{
  "status": "error",
  "errorType": "JOB_NOT_FOUND"
}
```

this message tell us that the “job” key is missing

SWire response: an example with “error_in_job” status

Request

```
{
  "job": [
    {
      "method": "com.stata.sfi.Data.getVarCount"
    },
    {
      "method": "com.stata.sfi.Data.notExistingMethod"
    }
  ]
}
```

intentionally wrong

Response

atomic job response with error

```
{
  "status": "error_in_job",
  "output": [
    {
      "status": "ok",
      "output": 12
    },
    {
      "status": "error",
      "errorType": "STATA_METHOD_NOT_FOUND"
    }
  ]
}
```

Appendix C

SWire special methods

$appendRow

Description

Append a record to the Stata dataset.

args

data – a map describing the data to be appended to the Stata dataset. In this map the keys are the variable names and the values are the corresponding values to be appended to those variables.

Output

No output.

Example

Request:

```
{
  "job": [{
    "method": "$appendRow",
    "args": {
      "data": {
        "var1": 21.3,
        "var2", 5.1,
      }
    }
  ]}
}
```

Response:

```
{
  "status": "ok",
  "output": [{
    "status": "ok"
  ]}
}
```

$generateVars

Description

Generate a set of Stata variables.

args

vars – an array of variable descriptions, where each variable description is a map in which the “name” key is the Stata variable name and the “type” key is the the Stata variable type (“byte”, “double”, “float”, “int”, “long”, “str” or “strl”).

Output

No output.

Example

Request:

```
{
  "job": [{
    "method": "$generateVars",
    "args": {
      "vars": [
        {"name": "var1", "type": "double"},
        {"name": "var2", "type": "int"},
      ]
    }
  ]}
}
```

Response:

```
{
  "status": "ok",
  "output": [{
    "status": "ok"
  ]}
}
```

$getData

Description

Get data (scalar or matrix) from Stata. This method is available from version 0.2.

args

data – an array where each element is a map with two properties: name, for the name of the data, and type, which can be "scalar" or "matrix".

Output

A map with a property called "data". This property contains the data retrieved from Stata.

Example

Request:

```
{
  "job": [{
    "method": "$getData",
    "args": {
       "data": [
         { "name": "x", "type": "scalar"},
         { "name": "y", "type": "matrix"}
       ]
     }
  }]
}
```

Response:

```
{
  "status": "ok",
  "data": [
    21,
    {
      "row": 1,
      "cols": 3,
      "data": [1, 2, 3]
    }
  ]
}
```

$getMatrix

Description

Obtain data from a Stata matrix, together with the number of rows and columns.

args

name – the matrix name

Output

A map whith the following nodes:
- rows: the number of matrix rows
- cols: the number of matrix columns
- data: an array of the matrix elements

Example

Request:

```
{
  "job": [{
    "method": "$getMatrix",
    "args": {
      "name": "foo"
    }
  ]}
}
```

Response:

```
{
  "status": "ok",
  "output": [{
    "status": "ok",
    "output": {
      "rows": 2,
      "cols": 3,
      "data": [11, 12, 13, 21, 22, 23]
    }
  ]}
}
```

$getNumericVar

Description

Obtain data from a numeric Stata variable.

args

name – the name of the requested numeric Stata variable.

Output

An array with the values of the requested variable.

Example

Request:

```
{
  "job": [{
    "method": "$getNumericVar",
    "args": {
      "name": "myvar"
    }
  ]}
}
```

Response:

```
{
  "status": "ok",
  "output": [{
    "status": "ok",
    "output": [3.4, 7.9, 12.4, 4.9]
  ]}
}
```

$getScalars

Description

Obtain the values of a set of Stata scalars.

args

scalars – an array of numeric Stata scalars.

Output

A map where the keys are the requested scalar names and the values are the corresponding scalar values.

Example

Request:

```
{
  "job": [{
    "method": "$getScalars",
    "args": {
      "scalars": ["foo", "bar"]
    }
  ]}
}
```

Response:

```
{
  "status": "ok",
  "output": [{
    "status": "ok",
    "output": {
      "foo": 21.3,
      "bar": 5.1
    }
  ]}
}
```

$getVarNames

Description

Obtain the variable names of the current Stata datataset.

args

No args key is required.

Output

An array containining the variable names of the current Stata dataset.

Example

Request:

```
{
  "job": [{
    "method": "$getVarNames"
  ]}
}
```

Response:

```
{
  "status": "ok",
  "output": [{
    "status": "ok",
    "output": ["var1", "var2", "var3"]
  }]
}
```

$ping

Description

Ping the SWire server to test its reachability.

args

No args key is required.

Output

The string “hello”.

Example

Request:

```
{
  "job": [{
    "method": "$ping"
  ]}
}
```

Response:

```
{
  "status": "ok",
  "output": [{
    "status": "ok",
    "output": "hello"
  ]}
}
```

$setNumericVar

Description

Set the values of a numeric Stata variable.

args

No args key is required.

Output

An array containing the Stata variable names.

Example

Request:

```
{
  "job": [{
    "method": "$setNumericVar",
    "args": {
      "name": "foo",
      "data": [1, 3, 5, 7, 9, 11, 13, 17]
    }
  ]}
}
```

Response:

```
{
  "status": "ok",
  "output": [{
    "status": "ok"
  ]}
}
```

$updateRowWithCredentials

Description

Update the Stata dataset row where the value of the username Stata variable and that of the password Stata variable correspond to those supplied in the command.

args

username – the username
password – password
data – a map where the keys are the variables to update and the values are the corresponding values.

Output

No output.

Example

Request:

```
{
  "job": [{
    "method": "$updateRowWithCredentials",
    "args": {
      "username": "johndoe",
      "password": "ah3kl$pk23c",
      "data": {
        "price": 5.6,
        "quantity": 24
      }
    }
  ]}
}
```

Response:

```
{
  "status": "ok",
  "output": [{
    "status": "ok"
  ]}
}
```

Appendix D

SWire4js

Introducing SWire4js

What is SWire4js?

SWire4js is a JavaScript library which facilitates the development of SWire web applications. It is self-contained in the file swire4js.js.

What does SWire4js provide?

- it encodes SWire requests and decodes SWire responses
- a utility function for generating Stata missing values

Requirements

msgpack-lite (http://kawanet.github.io/msgpack-lite/)

Project web page

https://sourceforge.net/projects/swire4js/

The SWire4js functions

- `swire.encode(jsonRequest)`
  Encodes a SWire JSON request
- `swire.decode(base64String)`
  Decodes a base64 String to a SWire JSON response
- `swire.getMissingValue(value)`
  Gets a Stata missing value

Getting started with SWire4js

test.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Test</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="js/msgpack.min.js"></script>
        <script src="js/swire4js.js"></script>
    </head>
    <body>
        <script>alert(swire.getMissingValue());</script>
    </body>
</html>
```

Output

An alert window with the value 8.98846567431158e+307

Required JavaScript files:

- msgpack.min.js (download from https://github.com/kawanet/msgpack-lite)
- swire4js.js (download from https://sourceforge.net/projects/swire4js/)

Encoding a SWire request

The enconding process:

```
{
  "job":[{"method":"com.stata.sfi.Data.getVarCount"}]
}
```

MessagePack serialization

```
81 A3 6A 6F 62 91 81 A6 6D 65 74 68 6F 64 BE 63 6F 6D 2E 73 74 61 74
61 2E 73 66 69 2E 44 61 74 61 2E 67 65 74 56 61 72 43 6F 75 6E 74
```

base64 encoding

```
gaNqb2KRgaZtZXRob2S+Y29tLnN0YXRhLnNmaS5EYXRhLmdldFZhckNvdW50
```

How you encode with SWire4js:

```
swire.encode({job:[{method:'com.stata.sfi.Data.getVarCount'}]});
// returns "gaNqb2KRgaZtZXRob2S+Y29tLnN0YXRhLnNmaS5EYXRhLmdldFZhckNvdW50"
```

Note: SWire4js internally uses msgpack-lite (http://kawanet.github.io/msgpack-lite/)

An example of a SWire web applications developed with SWire4js

```
$(document).ready(function() {
    var request = {job: [{method: 'com.stata.sfi.Data.getVarCount'}]};

    $.ajax({
        url: 'http://localhost:50000',
        data: swire.encode(request),
        method: "POST",
        success: function(data) {
            var response = swire.decode(data);
            if (response.status === 'ok')
                alert('Number of variables: ' + response.output[0].output);
            else
                alert('Error');
        },
        error: function() {
            alert('Network error');
        }
    });
});
```

Note: JQuery (https://jquery.com/) is required for this example

A minimal SWire web application

Local network

Request:

```
{
  "job": [{
      "method": "com.stata.sfi.Data.getObsCount"
  }]
}
```

Response:

```
{
  "status": "ok",
  "output" : [{
    "status": "ok",
    "output": 74
  }]
}
```

A minimal SWire web application: the HTML file

minimal.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>Minimal SWire web application</title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="js/jquery.js"></script>
        <script src="js/msgpack.min.js"></script>
        <script src="js/swire4js.js"></script>
        <script src="js/minimal.js"></script>
    </head>
    <body>
        <p>Obs count: <span id="count">?</span></p>
    </body>
</html>
```

A minimal SWire web application: the JavaScript file

minimal.js

```
$(document).ready(function() {
    var request = {
        job: [
            {
                method: 'com.stata.sfi.Data.getObsCount'
            }
        ]
    };

    $.ajax({
        url: 'http://127.0.0.1:50000',
        data: swire.encode(request),
        method: "POST",
        success: function(data) {
            var response = swire.decode(data);
            if (response.status === 'ok')
                $('#count').text(response.output[0].output);
        },
        error: function() {
            alert('network error');
        }
    });
});
```

Reference: swire.encode

```
swire.encode(jsonRequest)
```

Description

Encode a JSON request.

Parameters

jsonRequest – a JSON object representing a SWire request.

Returns

Nothing.

Examples

```
swire.encode({job:[{method:'com.stata.sfi.Data.getVarCount'}]});
// returns "gaNqb2KRgaZtZXRob2S+Y29tLnN0YXRhLnNmaS5EYXRhLmdldFZhckNvdW50"
```

Reference: swire.decode

```
swire.decode(base64String)
```

Description

Decode a base64 response string.

Parameters

base64String – a base64 encoded string representing a SWire response.

Returns

A JSON.

Examples

```
swire.decode('gqZvdXRwdXSRgqZzdGF0dXOib2umb3V0cHV0DKZzdGF0dXOib2s=');
// returns {status:'ok',output:[{status:'ok',output:12}]}
```

Reference: swire.getMissingValue

```
swire.getMissingValue(value)
```

Description

Get a Stata missing value.

Parameters

value – an “a” to “z” optional string, representing a Stata missing value. If *value* is not provided, then a standard missing value (.) will be returned.

Returns

A number representing the requested Stata missing value.

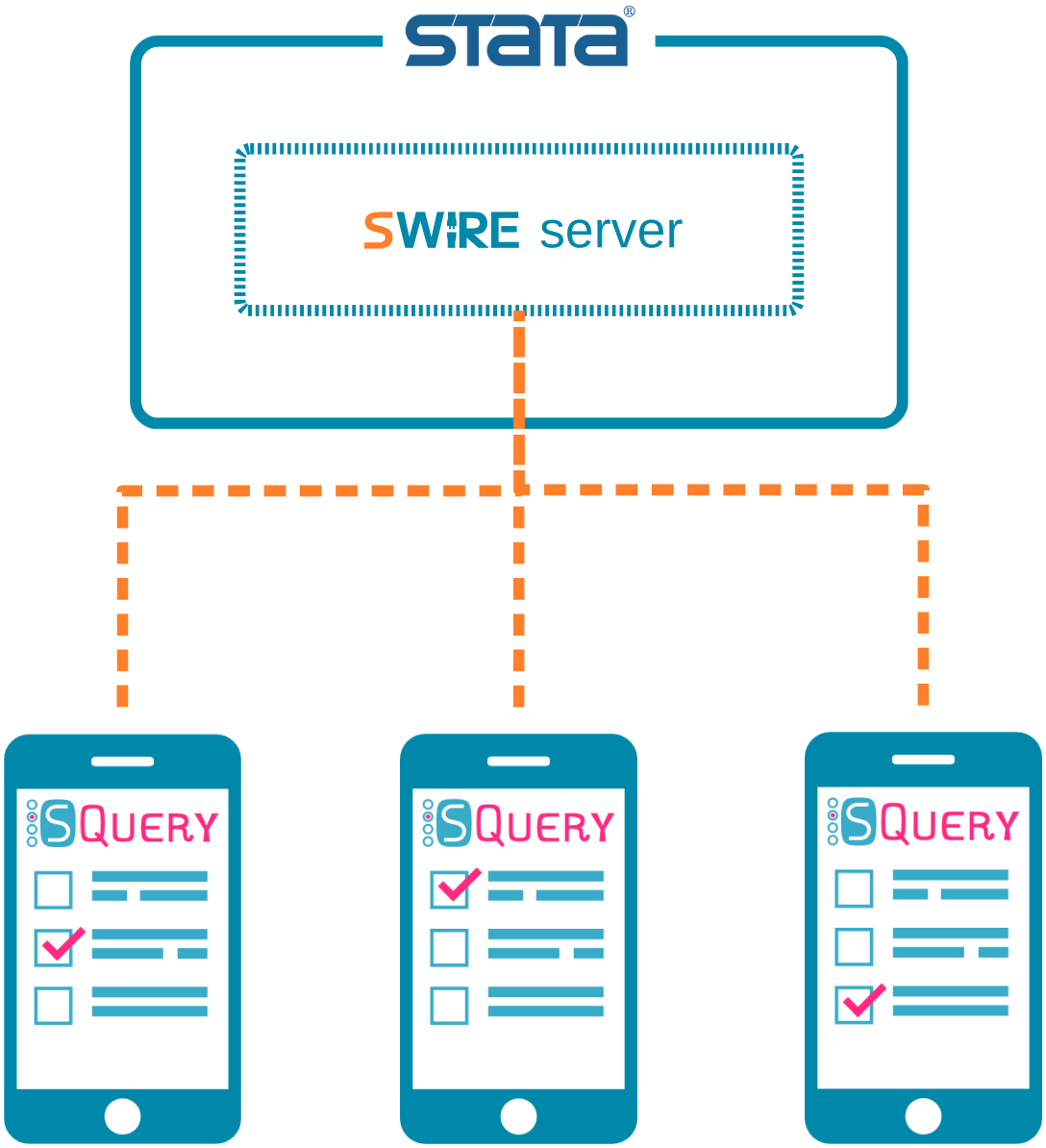
Examples

```
swire.getMissingValue(); // returns 8.98846567431158e+307
swire.getMissingValue('g'); // returns 9.003826821704202e+307
```

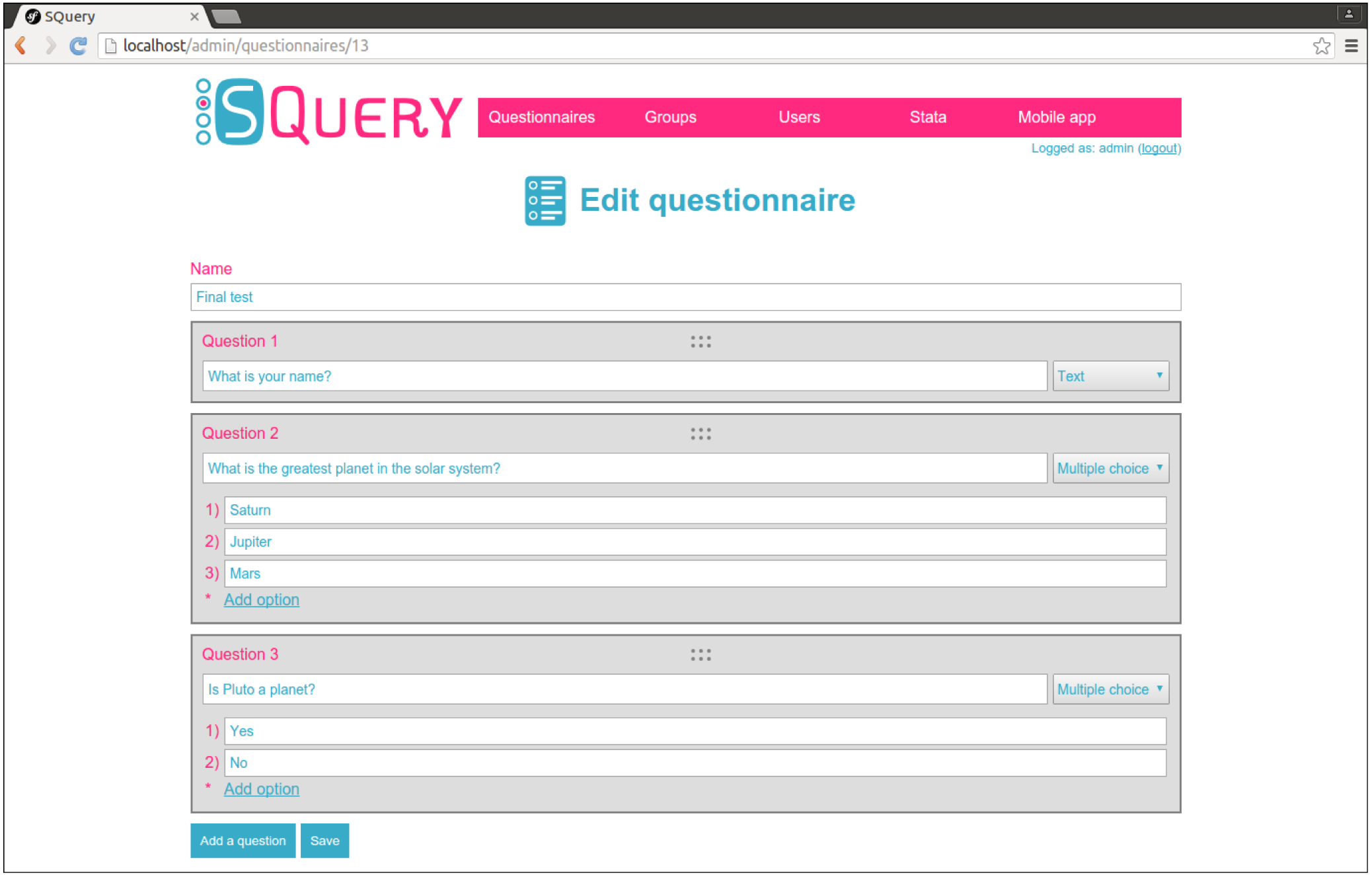
Appendix E

SQuery at a glance

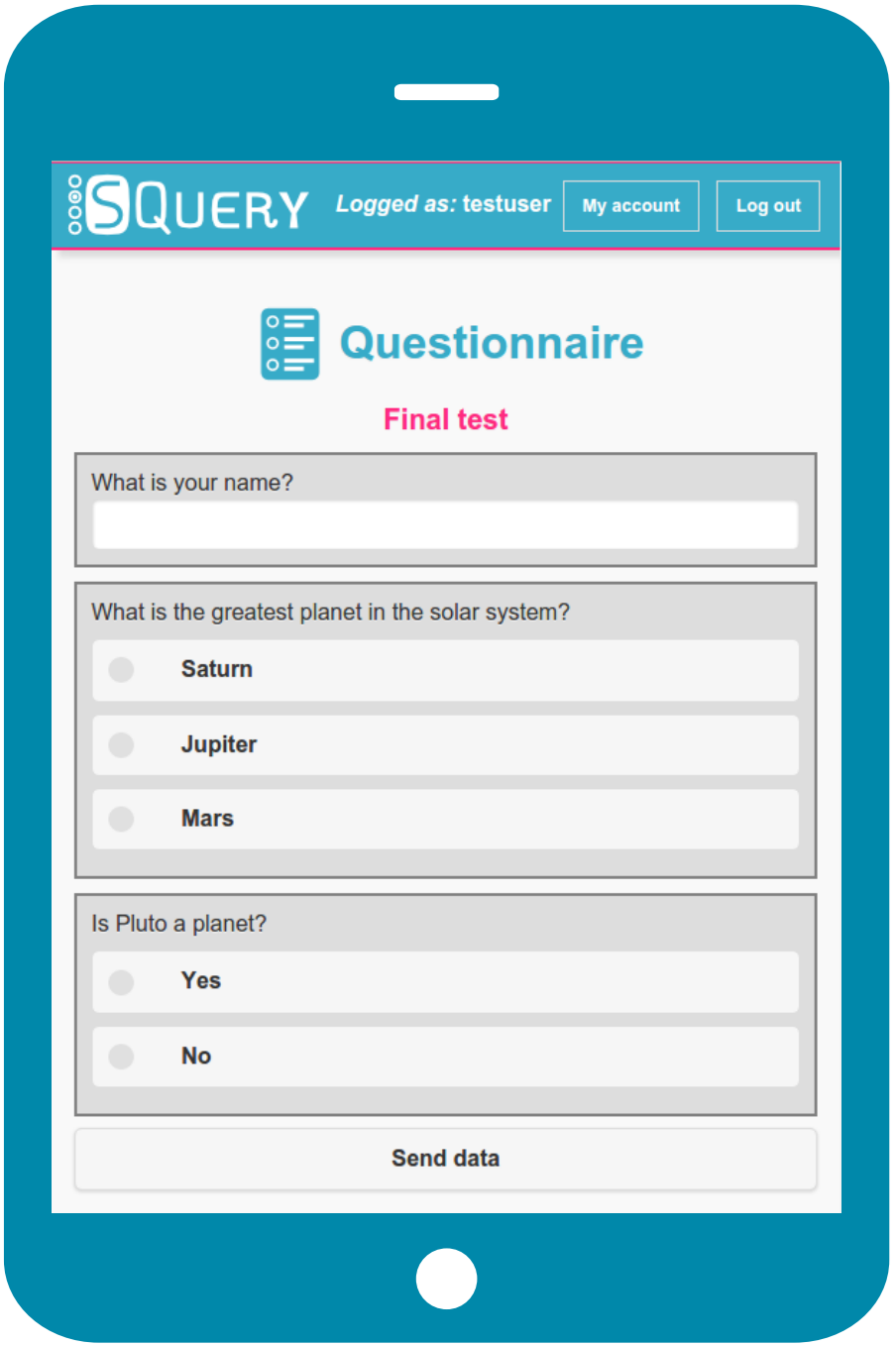
- It can create questionnaires and publish them on a local network
- It can collect responses from mobile devices
- Users can complete the questionnaire from their web browser
- Data are directly stored in Stata



Creating a questionnaire



Users can complete the questionnaire with their mobile devices



The starting Stata dataset

SQuery creates a Stata dataset where each record is a user's questionnaire

	q_1	q_2	q_3	username	password
1.		.	.	jack	$2y$12$49v/cB9Zmmy0zFzmTD.nfegcu3/m2.Ah9Em0Wa/XNVA3zCLRloGoi
2.		.	.	john	$2y$12$o8iY0nROvV6gGn8ueS7sHuN/Jdci3jgz8j0fDxKxwpMaude/vh2wy
3.		.	.	mary	$2y$12$uIj2XlWNEQtc1C3tVEuHnOCQuFqf3Q8PCqCWlTEEFpyUD84l4Gx/W
4.		.	.	paul	$2y$12$.TqtADrtvQHW4RmwWzS4WuCgQOUEFxT.bV05R8uLDdoZyoBVTStuq
5.		.	.	testuser	$2y$12$8gyxLCViiYBikdWFRF9NkeEaEC/oc8XeCUGGCKil7ydndouS/iZ6G

- *q_1*, *q_2* and *q_3* are items in the questionnaire
- the username and password variables are required to grant the permission to write in the record
- passwords are encrypted for privacy reasons

The final Stata dataset

Users complete the questionnaire:

	q_1	q_2	q_3	username	password
1.	Jack Pearson	1	1	jack	$2y$12$49v/cB9Zmmy0zFzmTD.nfegcu3/m2.Ah9Em0Wa/XNVA3zCLRloGoi
2.	John Lackland	2	1	john	$2y$12$o8iY0nROvV6gGn8ueS7sHuN/Jdci3jgz8j0fDxKxwpMaude/vh2wy
3.	Mary Blood	2	2	mary	$2y$12$uIj2XlWNEQtc1C3tVEuHnOCQuFqf3Q8PCqCWlTEEFpyUD84l4Gx/W
4.	Paul Jackson	3	2	paul	$2y$12$.TqtADrtvQHW4RmwWzS4WuCgQOUEFxT.bV05R8uLDdoZyoBVTStuq
5.	John Doe	1	2	testuser	$2y$12$8gyxLCViiYBikdWFRF9NkeEaEC/oc8XeCUGGCKil7ydndouS/iZ6G

data from users

...and now you can process your data in Stata

Other features of SQuery

- users can be organized into groups
- users can autonomously sign-in and the admin can create new users
- the admin can allow or forbid a sign-in to SQuery
- the admin sets the active questionnaire and the active group
- the front-end is mobile-optimized

Installation

Requirements

- MySQL
- PHP >= 5.4
- Composer (https://getcomposer.org/)
- A web server

Instructions and download

https://sourceforge.net/projects/squery-project/

Technical details about SQuery

- It is a web application
- It has been designed for use on a local network
- It has been developed with Symfony
- Composer is required for installation


SWIRE