Part D. More on EpiData software

Part D: More on EpiData software

Exercise 1: Relational database and aggregating vs from “Long-to-wide”

Exercise 2: A statistical process control chart

Exercise 3: A simplified survival analysis

Exercise 4: Creating a menu for standard reports

Exercise 5: Formatting standardized analysis output in a spreadsheet

Introductory note

Part D will address operationally relevant concepts in data collection and data analysis:

- How do we deal with a situation, where each individual has a varying number of observations?
- How do we determine statistically relevant deviations from a proportion over an observation period when the denominator varies with each time unit over the observation period?
- What is survival analysis and how to deal with it in EpiData Analysis?
- How to make a menu-driven, `HTML`-based EpiData Analysis interface to run standard reports?

Exercise 1: A relational database and "Aggregating" vs from "long-to-wide"

At the end of this exercise you should be able to:

a. Create a relational database for a varying number of observations
b. Merge a child file to the parent file
c. Recognizing when to use "Aggregate" and when to transpose data
d. Transpose multiple observations (columns) into a single record (row)

Not all laboratories keep their registers as The Union and WHO recommend for the Tuberculosis Laboratory Register, where 1 line corresponds to 1 examinee rather than to 1 examination. In fact, many laboratories enter the results for each examination on one line. If such an approach is chosen, we may find a register as follows:

Patient	Date of exam	Sex	Marital status	Blood sugar	Sputum	Result
A	24-Mar-2007	Male	Married	6.3	Mucoid	1+
B	24-Mar-2007	Male	Divorced	4.9	Muco-purulent	Neg
C	24-Mar-2007	Female	Single	5.2	Purulent	Neg
D	24-Mar-2007	Female	Widowed	7.3	Blood-tinged	2 per 100
A	25-Mar-2007	Male		7.3	Salivary	Neg
D	25-Mar-2007	Female		7.4	Mucoid	2+
A	26-Mar-2007			7.2	Purulent	1+
C	26-Mar-2007	Female		4.8	Muco-purulent	Neg
E	27-Mar-2007	Male	Married	8.2		1+
F	27-Mar-2007	Female	Annulled	7.4	Purulent	Neg
G	27-Mar-2007	Male	Cohabitating	6.9	Salivary	Neg
G	28-Mar-2007	Male		7.2	Mucoid	2+
E	28-Mar-2007	Male		7.9	Purulent	2+
F	31-Mar-2007	Female		7.2	Muco-purulent	3+
H	31-Mar-2007		Married	6.6	Mucoid	Neg
I	31-Mar-2007	Male	Separated	8.3	Salivary	Neg
H	1-Apr-2007	Female		6.9	Muco-purulent	1+
F	1-Apr-2007	Female	Engaged	7.7	Purulent	2+
I	1-Apr-2007	Male	Single	8.0	Mucoid	8 per 100
G	1-Apr-2007	Male		7.6	Muco-purulent	1+
K	2-Apr-2007	Female	Married	4.5	Purulent	Neg
I	2-Apr-2007	Male		8.2	Muco-purulent	
H	2-Apr-2007	Female		6.6	Mucoid	1+
I	3-Apr-2007	Male		8.1	Mucoid	1+

This type of a register requires a different approach to both data entry and data analysis than we used before. Two important things need to be considered:

1) The same patient may appear again and again on sequential dates
2) Not every patient has the same number of visits

Some patient characteristics do not change over time such as, in this example, the identity of the patient, sex, and marital status (well, perhaps not during these short intervals). Others do change, such as the date of examination, blood sugar, the aspect of the sputum and the sputum smear examination result.

To capture such information in a single data entry form would be very inconvenient: 1) one would have to anticipate the maximum of allowable visits, and 2) if one patient has a single visit, one would still have to complete all fields with the codes for missing values up to the maximum allotted if we insist that all fields must be `MUSTENTER` fields.

> ***Rule:*** *If an individual has a single observation for each variable or a fixed number of observations for each variable, then a single EpiData entry form is the best solution. If an individual has a variable number of observations for each variable, the choice is a* ***relational database****.*

Building a relational database requires determining which information is static for an individual and which changes over repeated observations.

EpiData Manager and EntryClient

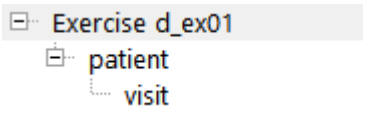
You may refer to Exercise 7 "Relational database" in Part A on the principles of and how to create a relational database.

The working example

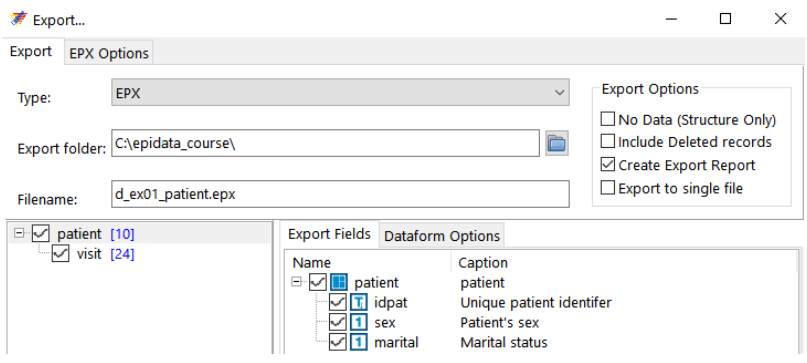
You will create an EpiData file:

`d_ex01.epx`

To get uniform naming during design and entry, we propose the following:



The project is saved as "d_ex01.epx" and the two databases it consists of are the parent dataset named "patient" and the child set "visit". When exporting:



EpiData will automatically create the appropriate two files:

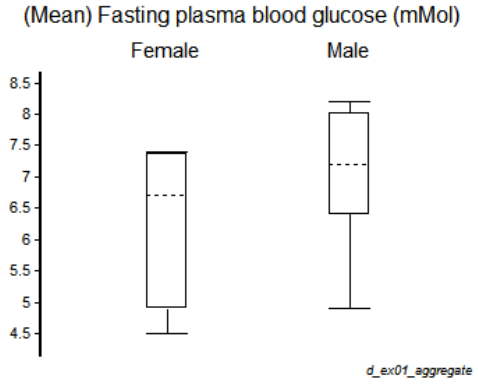
```
d_ex01_patient.epx
d_ex01_visit.epx
```

These files will be required in EpiData Analysis for merging.

EpiData Analysis

The final result of the analysis will be to obtain the following EpiData Analysis output:

Part 1 with AGGREGATE



Part 2 with transposing values

Overall microscopy result

of 4 serial smears	Negative	Positive	Total
N—	2	0	2
NN–	1	0	1
NN9P	0	1	1
NP–	0	1	1
NPP-	0	3	3
PNP-	0	1	1
PP–	0	1	1
Total	3	7	10

Patient's sex

Incremental yield of first 3 smears	Female	%	Male	%	Total	%
NNP	0	(0.0)	1	(25.0)	1	(14.3)
NPx	3	(100.0)	1	(25.0)	4	(57.1)
Px	0	(0.0)	2	(50.0)	2	(28.6)
Total	3	(100.0)	4	(100.0)	7	

Percents: (Col)

It doesn't really look like much. Nevertheless, quite a few steps are needed to get from the source files `D_EX01_PATIENT.EPX` and `D_EX01_VISIT.EPX` to this point. We will elaborate on some considerations you have to make and offer hints for the sequential components in EpiData Analysis.

Merging the files

In EpiData Manager you made a relation through a unique identifier from the parent to the child file. In EpiData Analysis you must now merge these to files to get the following dataset.

We start by reading the child file:

```
read "childfile.epx"
merge parentidentifier /file="parentfile.epx" /table
```

To each record from the child file, information from the parent file is added repetitively for each observation for the same individual. In other words, you start from the other way around than in EpiData EntryClient, starting in EpiData Analysis with the child file and using the parent as a lookup table.

As a result we get (first 13 records only shown):

	idpat	visitid	visitdate	bs	sputum	micres	sex	marital	MergeVar
1	A	A-25-03-2007	25/03/2007	7.3	Salivary	Negative	Male	Married	In both
2	A	A-26-03-2007	26/03/2007	7.2	Purulent	1+ positive	Male	Married	In both
3	A	A-24-03-2007	24/03/2007	6.3	Mucoid	1+ positive	Male	Married	In both
4	B	B-24-03-2007	24/03/2007	4.9	Muco-purulent	Negative	Male	Divorced	In both
5	C	C-24-03-2007	24/03/2007	5.2	Purulent	Negative	Female	Single	In both
6	C	C-26-03-2007	26/03/2007	4.8	Muco-purulent	Negative	Female	Single	In both
7	D	D-25-03-2007	25/03/2007	7.4	Mucoid	2+ positive	Female	Widowed	In both
8	D	D-24-03-2007	24/03/2007	7.3	Blood-tinged	Scanty positive	Female	Widowed	In both
9	E	E-28-03-2007	28/03/2007	7.9	Purulent	2+ positive	Male	Married	In both
10	E	E-27-03-2007	27/03/2007	8.2	Not recorded	1+ positive	Male	Married	In both
11	F	F-01-04-2007	01/04/2007	7.7	Purulent	2+ positive	Female	Annulled	In both
12	F	F-31-03-2007	31/03/2007	7.2	Muco-purulent	3+ positive	Female	Annulled	In both

A variable MERGEVAR has been created by EpiData Analysis. It can take 3 values:

```
1 Only in memory (original)
2 Only in external file
3 In both
```

A frequency helps to identify quickly whether there are for instance any “orphans”, that is child records which do not find the same identifier in a parent record and are thus useless.

We note in the above that the visit dates are not temporally sequential within each parent identifier, we need thus sorting, first by parent identifier, then within them by date, thus:

```
sort idpat visitdate
```

However, this may not be correct in all circumstances.

Note: *We commonly suggested to code unknown dates as “01/01/1800”. With sorting, the unknown date will thus come first (sorting is ascending by default) and this might not be desirable. In this dataset we don’t have unknown dates. But only because the small dataset allows us to see that, you would have to anticipate that possibility with a larger dataset and thus first make a new date variable where the unknowns take a value for a date in the future, so they appear with sorting at the end of the information on an individual.*

Following the sorting, it might be advantageous to number the visits in order to be able to make a frequency on them to see what the maximum number of visits is (here we can see that it is a maximum of 4 visits, but in a large database, it would be more difficult).

To create a new variable EXAM and set its default initially to 1 (each record takes first the value 1), we have so far used the following grammar:

```
define exam #
exam=1
```

or alternatively the one-line alternative approach which accomplishes exactly the same thing:

```
gen i exam=1
```

> ***Note:*** *the command* `GEN` *will produce integer of length of 9. If you need a shorter and fixed field length integer, you must utilize* `DEFINE`*.*

The command `GEN` replaces `DEFINE` and "`i`" stands for an integer field.

There are other similar commands (see an earlier Exercise):

```
gen f doorheight=1.85
gen d birthdate=dmy(31,12,1899)
gen s firstname="john"
```

for date fields (`d`) float (real number) fields (`f`), and string (text) fields (`s`). Look it up in the help file (type `gen`+**F1** in the command line).

Now that you have a new variable `EXAM`, how can you tell EpiData that it should look at the person (identified by an `ID`) and number each visit, starting with 1 until the next individual comes, when it must start again with 1. The command is:

```
if id=id[_n-1] then visit=visit[_n-1]+1
```

This looks admittedly complex. Let's thus take it apart. `[_n]` identifies the current record and accordingly `[_n-1]` the immediately preceding record. Let's say, EpiData Analysis has proceeded to record 547 and looks at the `ID` of this record. It looks whether record 546 had the same `ID` as record number 547: `if id=id[_n-1]`. If that is the case, then it should take the `VISIT` number of record 546 and add 1 to it for record 547: `visit=visit[_n-1]+1`. If it is not the case, then the default stays (which we defined as `1`) and it moves on to the next record.

As a result, you should get:

	idpat	visitdate	visit
1	A	24/03/2007	1
2	A	25/03/2007	2
3	A	26/03/2007	3
4	B	24/03/2007	1
5	C	24/03/2007	1
6	C	26/03/2007	2
7	D	24/03/2007	1
8	D	25/03/2007	2
9	E	27/03/2007	1
10	E	28/03/2007	2
11	F	27/03/2007	1
12	F	31/03/2007	2
13	F	01/04/2007	3

the first variable column showing the identifier, the second the date of the examination, and the third the number of the examination for that individual.

Aggregating data

If we look at the sex (given the field name SEX), date of visit (given the field name VISITDATE), and blood sugar (given the field name BS):

	idpat	sex	visitdate	bs
1	A	Male	24/03/2007	6.3
2	A	Male	25/03/2007	7.3
3	A	Male	26/03/2007	7.2
4	B	Male	24/03/2007	4.9
5	C	Female	24/03/2007	5.2
6	C	Female	26/03/2007	4.8

SEX remains obviously the same (and is thus from the parent file), and is a categorical variable unique to the examined person, while BS varies by examination date and is a continuous variable. If we want to examine blood sugar by sex, there is no need to transpose the blood sugar values from the vertical to the horizontal as EpiData Analysis has inbuilt a tool to aggregate the data. For the individual and its sex we would write:

```
aggregate idpat sex /close
```

and get with BROWSE the ten individuals and their SEX:

	idpat	sex	N
1	A	Male	3
2	B	Male	1
3	C	Female	2
4	D	Female	2
5	E	Male	2
6	F	Female	3
7	G	Male	3
8	H	Female	3
9	I	Male	4
10	K	Female	1

We can expand this command using an option to calculate the MEAN of blood sugar for each individual:

```
aggregate idpat sex /mean="bs" /close
```

and get:

	idpat	sex	N	Nbs	MEAbs
1	A	Male	3	3	6.9333333333
2	B	Male	1	1	4.9000000000
3	C	Female	2	2	5.0000000000
4	D	Female	2	2	7.3500000000
5	E	Male	2	2	8.0500000000
6	F	Female	3	3	7.4333333333
7	G	Male	3	3	7.2333333333
8	H	Female	3	3	6.7000000000
9	I	Male	4	4	8.1500000000
10	K	Female	1	1	4.5000000000

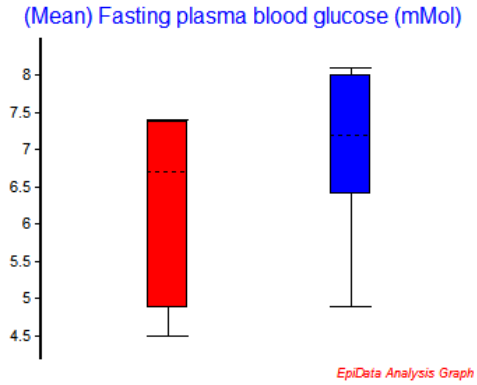
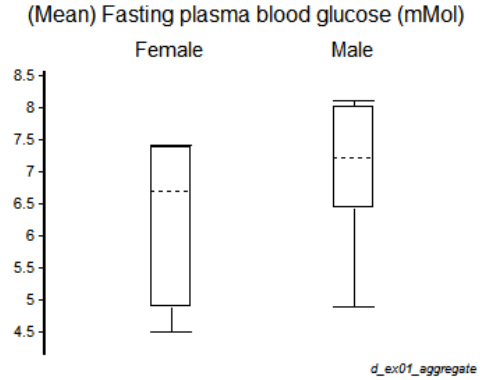
where MEAbs is the calculated mean value of blood sugar for an individual from all the individual's measurements. To save the aggregate data in a file, we add another option:

```
aggregate patid sex /mean="bs" /close /save="d_ex01_aggregate.rec" /replace
```

Having accomplished this, it is now straight forward to write:

```
cls
close
read "d_ex01_aggregate.rec"
boxplot meabs /by=sex
```

and get:

Default (shown above)	Using additional options (see Help file)
	

From long-to-wide

For showing means, there is thus no need to transpose data from the vertical to the horizontal. But it is different for the macroscopic aspect of sputum the examination results. We do not want (nor would we get a sensible result) aggregate sputum smear results. We wish to know each result from each individual.

The first thing before starting copying results from the vertical to the horizontal, we need to know the maximum number of examinations an individual had in the data set. We can get this with a frequency on the `VISIT`:

```
freq visit
```

Number of visit

	N
1	10
2	8
3	5
4	1
Total	24

This shows that the maximum number of examinations an individual had in this dataset was 4. We need thus to prepare 4 new variables for each field that are in the sequence of the examination in the vertical but should also become part of each record.

Let's say we have a variable `VAR1` with different values for each visit:

```
ID   VISIT   VAR1
B1   1       1
B1   2       3
B1   3       2
B1   4       2
C1   1       3
D1   1       2
```

What we need is:

```
ID   VISIT    VAR1      VAR11     VAR12      VAR13     VAR14
B1   1        1
B1   2        3
B1   3        6
B1   4        2
C1   1        3
D1   1        2
```

First, we make these 4 variables and give them all the default value of `-1` (the minus one is a good way to see missing data and helps later in the selection):

```
gen i var11=-1
gen i var12=-1
gen i var13=-1
gen i var14=-1
```

After these four command lines, our above dataset becomes:

```
ID   VISIT    VAR1      VAR11     VAR12      VAR13     VAR14
B1   1        1         -1        -1         -1        -1
B1   2        3         -1        -1         -1        -1
B1   3        6         -1        -1         -1        -1
B1   4        2         -1        -1         -1        -1
C1   1        3         -1        -1         -1        -1
D1   1        2         -1        -1         -1        -1
```

and we are ready to copy the values from the vertical to the horizontal by respecting that the value of `VAR1` from `VISIT 1` goes to `VAR11`, the value from `VISIT 2` to `VAR12`, etc.

For `VISIT 1`, the value for `VAR11` is equal to the value of `VAR1` and we make it thus the default:

```
VAR11=VAR1
```

Then we use the same approach as above to identify the record:

```
if id[_n]=id[_n+1) then var12=var1[_n+1]
```

This means that if the current record `[_n]` has the same `ID` as the next record `[_n+1]`, then `VAR12` in the current record should take the value of `VAR1` from the next record `[_n+1]`. Now we do this for all possible 4 records (the maximum of `VISIT`s):

```
if id[_n]=id[_n+2] then var13=var1[_n+2]
if id[_n]=id[_n+3] then var14=var1[_n+3]
```

All the lines to be written for this original field `VAR1` are thus:

```
gen i var11=-1
gen i var12=-1
gen i var13=-1
gen i var14=-1
VAR11=VAR1
if id[_n]=id[_n+1] then var12=var1[_n+1]
if id[_n]=id[_n+2] then var13=var1[_n+2]
if id[_n]=id[_n+3] then var14=var1[_n+3]
```

and we get (assuming that the last patient `D1` had only 1 `VISIT`):

```
ID   VISIT   VAR1    VAR11    VAR12    VAR13    VAR14
B1   1       1       1        3        6        2
B1   2       3       1        3        6        -1
B1   3       6       1        3        -1       -1
B1   4       2       1        -1       -1       -1
C1   1       3       3        -1       -1       -1
D1   1       2       2        -1       -1       -1
```

You may note that only for `VISIT` 1 for patient `B1` with four visits all new 4 variables have all respective 4 values from the 4 `VISIT`s.

Now we can safely get rid of the records of `VISIT`s 2, 3, and 4 and we end up just with individuals who have all information from each `VISIT`:

```
select visit=1
```

and we get:

```
ID   VISIT   VAR1    VAR11    VAR12    VAR13    VAR14
B1   1       1       1        3        6        2
C1   1       3       3        -1       -1       -1
D1   1       2       2        -1       -1       -1
```

The conversion from "Long-to-wide" is thus successfully completed, well, for this variable. Of course, before you make this selection, you have to repeat the same approach for each field, so that in the end you get something like:

	idpat	sex	marital	visitdate1	visitdate2	visitdate3	visitdate4	sputum1	sputum2	sputum3	sputum4	micres1	micres2	micres3	micres4	pattern	case	yield
1	A	Male	Married	24/03/2007	25/03/2007	26/03/2007		Mucoid	Salivary	Muco-purulent		1+ positive	Negative	1+ positive		PNP	Positive	Px
2	B	Male	Divorced	24/03/2007				Purulent				Negative				N	Negative	
3	C	Female	Single	24/03/2007	26/03/2007			Muco-purulent	Purulent			Negative	Negative			NN	Negative	
4	D	Female	Widowed	24/03/2007	25/03/2007			Blood-tinged	Mucoid			Scanty positive	1+ positive			PP	Positive	Px
5	E	Male	Married	27/03/2007	28/03/2007			Not recorded	Muco-purulent			1+ positive	1+ positive			PP	Positive	Px
6	F	Female	Annulled	27/03/2007	31/03/2007	01/04/2007		Muco-purulent	Purulent	Muco-purulent		Negative	1+ positive	1+ positive		NPP	Positive	NPx
7	G	Male	Cohabitating	27/03/2007	28/03/2007	01/04/2007		Salivary	Mucoid	Purulent		Negative	1+ positive	1+ positive		NPP	Positive	NPx
8	H	Female	Married	31/03/2007	01/04/2007	02/04/2007		Mucoid	Purulent	Mucoid		Negative	1+ positive	1+ positive		NPP	Positive	NPx
9	I	Male	Separated	31/03/2007	01/04/2007	02/04/2007	03/04/2007	Salivary	Mucoid	Purulent	Mucoid	Negative	Scanty posi	Not recorde	1+ positive	NP9P	Positive	NPx
10	K	Female	Married	02/04/2007				Muco-purulent				Negative				N	Negative	

You have now ten patients and you are at the point where you can continue to work in the same way as you used to work before. While it is much more complex to get to here from 1 line per examination than from 1 line per examinee, it is also obvious that in the end this is much more informative.

For each examination we have the date and for each specimen we have the quality of the sputum. In the Union / WHO approach you have only one date (the date of collection of the first specimen) for a series of three, and the quality of sputum in the Tuberculosis Laboratory is not that informative as it is very possible that every day the quality of the specimen is different, but there is no space allocated to write 3 different ones.

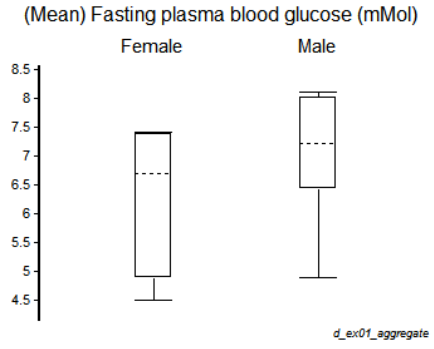
Tasks:

- ***Prepare a data documentation sheet***
- ***Prepare the EpiData Manager form for the relational database***
- ***Enter the data from the following sample data set:***

Patient	Date of exam	Sex	Marital status	Blood sugar	Sputum	Result
A	24-Mar-2007	Male	Married	6.3	Mucoid	1+
B	24-Mar-2007	Male	Divorced	4.9	Muco-purulent	Neg
C	24-Mar-2007	Female	Single	5.2	Purulent	Neg
D	24-Mar-2007	Female	Widowed	7.3	Blood-tinged	2 per 100
A	25-Mar-2007	Male		7.3	Salivary	Neg
D	25-Mar-2007	Female		7.4	Mucoid	2+
A	26-Mar-2007			7.2	Purulent	1+
C	26-Mar-2007	Female		4.8	Muco-purulent	Neg
E	27-Mar-2007	Male	Married	8.2		1+
F	27-Mar-2007	Female	Annulled	7.4	Purulent	Neg
G	27-Mar-2007	Male	Cohabitating	6.9	Salivary	Neg
G	28-Mar-2007	Male		7.2	Mucoid	2+
E	28-Mar-2007	Male		7.9	Purulent	2+
F	31-Mar-2007	Female		7.2	Muco-purulent	3+
H	31-Mar-2007		Married	6.6	Mucoid	Neg
I	31-Mar-2007	Male	Separated	8.3	Salivary	Neg
H	1-Apr-2007	Female		6.9	Muco-purulent	1+
F	1-Apr-2007	Female	Engaged	7.7	Purulent	2+
I	1-Apr-2007	Male	Single	8.0	Mucoid	8 per 100
G	1-Apr-2007	Male		7.6	Muco-purulent	1+
K	2-Apr-2007	Female	Married	4.5	Purulent	Neg
I	2-Apr-2007	Male		8.2	Muco-purulent	
H	2-Apr-2007	Female		6.6	Mucoid	1+
I	3-Apr-2007	Male		8.1	Mucoid	1+

- ***Write a program D_EX01.PGM that merges the two files, then prepare sets for the aggregated data and for the "long-to-wide" transformation to produce the following output respectively:***

From aggregating the data:



From transformation "long-to-wide":

"Table 1. Pattern of serial smear results"

Definition of case by microscopy of 4 serial smears	Negative	Positive	Total
N---	2	0	2
NN--	1	0	1
NP?P	0	1	1
NPP-	0	3	3
PNP-	0	1	1
PP--	0	2	2
Total	3	7	10

"Table 2. Incremental yield among positive results"

Incremental yield of first 3 smears	Patient's sex: Female	%	Male	%	Total	%
NPx	2	(66.7)	2	(50.0)	4	(57.1)
Px	1	(33.3)	2	(50.0)	3	(42.9)
Total	3	(100.0)	4	(100.0)	7	

Percents: (Col)