## Pairwise Overview

Pairwise is a testing technique used to maximize test coverage while minimizing test cases. Since testing all possible combinations of input parameters would require a Cartesian product of tests, it isn't reasonable, if even possible, to do. Instead, but why when you have pairwise?

Most defects trigger with either a single input parameter or an interaction between pairs of parameters. So, testing all possible combinations of all pairs of input parameters can only be done by using a Cartesian product of tests, but why when you have pairwise?

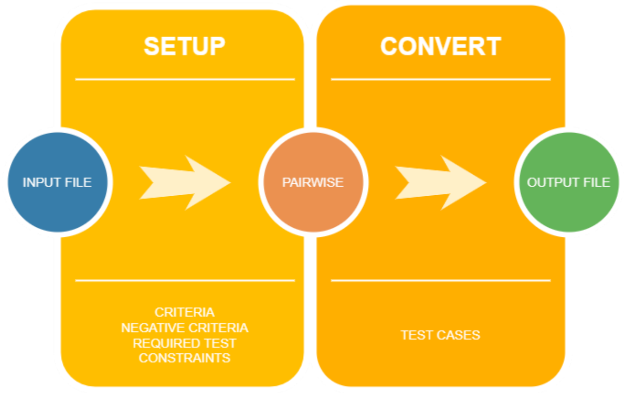
Pairwise makes it possible to save time and money by **significantly** reducing the number of tests needed to test.

To make it more useful, AnkrPt Pairwise not only generates all possible pairs of input parameters, it also lets you define required tests, negative criteria, and constraints to achieve a well-designed and complete test suite.

Pairwise testing should be an integral part of every test plan, making testing more effective with fewer tests thus reducing test estimates. In addition, data test sets created with Pairwise can be reused or easily modified during development which needs retesting.

## Process Overview

Pairwise process diagram:



## Example Criteria Files

The following samples are for illustration purposes and show Criteria, Negative Tests, Required Tests, and Constraints. Use these as a starting point when desired.

In the examples that blank lines are ignored, and spaces before and after values are also ignored.

Duplicating values in the same criteria is not allowed so blank characters do not make them distinct. But mixing case *does* make values distinct.

Buy-sell (criteria)

This is a basic buy-sell pairwise generation example found in many places on the internet. We used this as the basis to learn pairwise and validate our process.

```
CRITERIA
Sale,    Buy,     Sell
City,    Austin,  Gdansk
Status,  New,     Used
Vehicle, BMW,     Audi,  Mercedes
Color,   Red,     Blue
Hours,   Working, Non-working
```

Buy-sell (with negative criteria)

This is simple negative criteria. One negative test will be produced for each negative criteria.

```
CRITERIA
Sale                ,Buy    ,Sell
City                ,Austin ,Gdansk
Status              ,New    ,Used
Vehicle             ,BMW    ,Audi       ,Mercedes
Color               ,Red    ,Blue
Hours               ,Working,Non-working

NEGATIVE CRITERIA
Sale                ,Trade
Vehicle             ,Lada
Color               ,Yellow
```

Buy-sell (with required tests)

Required tests allow you to create specific conditions beyond pairs to be tested. These are always the first tests generated to use the specified pairs, and to allow you to easily confirm they're generated.

Test 4 does not provide all values. In these cases, the missing values will be filled in from the list of valid values.

```
CRITERIA
Sale                ,Buy    ,Sell
City                ,Austin ,Gdansk
Status              ,New    ,Used
Vehicle             ,BMW    ,Audi       ,Mercedes
Color               ,Red    ,Blue
Hours               ,Working,Non-working

```

```
REQUIRED TESTS
ID                ,Sale   ,City       ,Status   ,Vehicle,Color,Hours
1                 ,Buy    ,Austin     ,Used     ,Audi   ,Blue ,Non-working
2                 ,Buy    ,Gdansk     ,Used     ,Audi   ,Blue ,Working
3                 ,Sell   ,Gdansk     ,New      ,Audi   ,Red  ,Working
4                 ,       ,Gdansk     ,         ,BMW    ,Red
```

Buy-sell (with constraints)

Constraints are the basis of limiting value combinations, they are critical in generating useful tests.

```
CRITERIA
Sale              ,Buy    ,Sell
City              ,Austin ,Gdansk
Status            ,New    ,Used
Vehicle           ,BMW    ,Audi       ,Mercedes
Color             ,Red    ,Blue
Hours             ,Working,Non-working

CONSTRAINTS
When              ,Vehicle,is         ,BMW
And               ,City   ,is         ,Austin
Then              ,Color  ,isNot      ,Red
```

Buy-sell (with all sections)

This simple puts all the sections into a single file. Note that blank lines allow for section separation without impacting the generation process.

```
CRITERIA
Sale              ,Buy    ,Sell
City              ,Austin ,Gdansk
Status            ,New    ,Used
Vehicle           ,BMW    ,Audi       ,Mercedes
Color             ,Red    ,Blue
Hours             ,Working,Non-working

NEGATIVE CRITERIA
Sale              ,Trade
Vehicle           ,Lada
Color             ,Yellow

REQUIRED TESTS
ID                ,Sale   ,City       ,Status   ,Vehicle,Color,Hours
1                 ,Buy    ,Austin     ,Used     ,Audi   ,Blue ,Non-working
2                 ,Buy    ,Gdansk     ,Used     ,Audi   ,Blue ,Working
3                 ,Sell   ,Gdansk     ,New      ,Audi   ,Red  ,Working
4                 ,       ,Gdansk     ,         ,BMW    ,Red

CONSTRAINTS
When              ,Vehicle,is         ,BMW
And               ,City   ,is         ,Austin
Then              ,Color  ,isNot      ,Red
```

Web site (criteria, required tests, constraints)

```
CRITERIA
Device, Mobile, Desktop
Desktop Browser, Chrome, Firefox, Internet Explorer, Opera, Safari, n/a
```

```
Mobile Browser, Chrome, Android, Opera, Safari, n/a
Desktop Connection, wifi, cable, n/a
Mobile Connection, edge, 4g, wifi, n/a
Desktop OS, Windows, Linux, Mac, n/a
Mobile OS, iOS, Android, n/a
IP Version, IPv4, IPv6
Protocol, http, https
www in front, yes, no
Cookies, enabled, disabled
Window Dimension, wide, tall
Window Size, large, small
Language, english, non-english
Server Caching, on, off
Server Minification, on, off

REQUIRED TEST
ID, Desktop, Desktop Browser, Mobile Browser, Desctop Connection, Mobile Connection, Desktop OS, Mobile OS, IP Version, Protocol, WWW in front, Cookies, Window Dimension, Window Size, Language, Server Caching, Server Minification
Desktop, Desktop, Chrome, n/a, cable, n/a, Windows, n/a, IPv4, https, no, enabled, wide, large, english, on, off
Mobile, Mobile, n/a, Chrome, n/a, wifi, n/a, Android, IPv4, https, no, enabled, tall, small, english, on, off

CONSTRAINT
When, Device, IS, Mobile
Then, Mobile Browser, ISNOT, n/a
And, Mobile Connection, ISNOT, n/a
And, Mobile OS, ISNOT, n/a
And, Desktop Browser, IS, n/a
And, Desktop Connection, IS, n/a
And, Desktop OS, IS, n/a

When, Device, IS, Desktop
Then, Desktop Browser, ISNOT, n/a
And, Desktop Connection, ISNOT, n/a
And, Desktop OS, ISNOT, n/a
And, Mobile Browser, IS, n/a
And, Mobile Connection, IS, n/a
And, Mobile OS, IS, n/a

When, Desktop Browser, ISNOT, n/a
Or, Desktop Connection, ISNOT, n/a
Or, Desktop OS, ISNOT, n/a
Then, Mobile Browser, IS, n/a
And, Mobile Connection, IS, n/a
And, Mobile OS, IS, n/a

When, Mobile Browser, IS, Android
Then, Mobile OS, IS, Android

When, Desktop Browser, IS, Internet Explorer
Then, Desktop OS, IS, Windows

When, Mobile OS, ISNOT, n/a
Then, Desktop OS, IS, n/a

When, Mobile Browser, ISNOT, n/a
```

```
Then, Desktop Browser, IS, n/a

When, Mobile Connection, ISNOT, n/a
Then, Desktop Connection, IS, n/a

When, Device, IS, Mobile
Then, Window Size, IS, small
```

6x6 criteria only

This example is a basic alphabetic/number square to easily see and understand the results of generation. As with any criteria value, simply replacing one value with another doesn’t impact the generation process.

```
CRITERIA
a,a1,a2,a3,a4,a5,a6
b,b1,b2,b3,b4,b5,b6
c,c1,c2,c3,c4,c5,c6
d,d1,d2,d3,d4,d5,d6
e,e1,e2,e3,e4,e5,e6
f,f1,f2,f3,f4,f5,f6
```

# END USER LICENSE AGREEMENT

This copy of software ("the Software Product") and accompanying documentation is licensed and not sold. This Software Product is protected by copyright laws and treaties, as well as laws and treaties related to other forms of intellectual property. Kimputing, Inc. or its subsidiaries, affiliates, and suppliers (collectively "Licensor") own intellectual property rights in the Software Product. The Licensee’s ("you" or "your") license to download, use, copy, or change the Software Product is subject to these rights and to all the terms and conditions of this End User License Agreement ("Agreement").

## Acceptance

YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS AGREEMENT BY DOWNLOADING THE SOFTWARE PRODUCT OR BY INSTALLING, USING, OR COPYING THE SOFTWARE PRODUCT. IF YOU DO NOT AGREE TO ALL OF THE TERMS OF THIS AGREEMENT, YOU MUST NOT INSTALL, USE, OR COPY THE SOFTWARE PRODUCT.

## License Grant

This Agreement entitles you to a non-transferable license to use the Licensed Application as designed. You may not assign your rights and obligations under this Agreement, or redistribute, encumber, sell, rent, lease, sublicense, or otherwise transfer your rights to the Software Product. You will make no effort to decompile, "reverse-engineer", disassemble, or otherwise attempt to derive the source code for the Software Product. You may not modify the Software Product or create any derivative work of the Software Product or its accompanying documentation. Derivative works include but are not limited to translations. You may not alter any files or libraries in any portion of the Software Product.

## Consent to Use of Data

You agree that Licensor may collect and use technical data and related information, including but not limited to technical information about your device and system, that is gathered periodically to facilitate the provision of the software updates, product support and other services related to the Software Product. Licensor may use this information as long as it is in a form that does not personally identify you, to improve or market its products, services or technologies.

## Termination

This EULA is effective until terminated by you or Licensor. Your rights under this EULA will terminate automatically if you fail to comply with any of its terms.

## Disclaimer of Warranties and Limitation of Liability

YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT USE OF THE SOFTWARE PRODUCT IS AT YOUR SOLE RISK AND THAT THE SOFTWARE PRODUCT IS PROVIDED “AS IS,” WITH ALL FAULTS AND WITHOUT WARRANTY OF ANY KIND. LICENSOR MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN FACT OR IN LAW, INCLUDING, BUT NOT LIMITED TO, ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE OTHER THAN AS SET FORTH IN THIS AGREEMENT. LICENSOR MAKES NO WARRANTY THAT THE SOFTWARE PRODUCT WILL MEET YOUR REQUIREMENTS OR OPERATE UNDER YOUR SPECIFIC CONDITIONS OF USE.

YOU MUST DETERMINE WHETHER THE SOFTWARE PRODUCT SUFFICIENTLY MEETS YOUR REQUIREMENTS FOR SECURITY AND UNINTERRUPTABILITY. YOU BEAR SOLE RESPONSIBILITY AND ALL LIABILITY FOR ANY LOSS INCURRED DUE TO FAILURE OF THE SOFTWARE PRODUCT TO MEET YOUR REQUIREMENTS. LICENSOR WILL NOT, UNDER ANY CIRCUMSTANCES, BE RESPONSIBLE OR LIABLE FOR THE LOSS OF DATA ON ANY COMPUTER OR INFORMATION STORAGE DEVICE.

UNDER NO CIRCUMSTANCES SHALL LICENSOR, ITS DIRECTORS, OFFICERS, EMPLOYEES OR AGENTS BE LIABLE TO YOU OR ANY OTHER PARTY FOR INDIRECT, CONSEQUENTIAL, SPECIAL, INCIDENTAL, PUNITIVE, OR EXEMPLARY DAMAGES OF ANY KIND (INCLUDING LOST REVENUES OR PROFITS OR LOSS OF BUSINESS) RESULTING FROM THIS AGREEMENT, OR FROM THE FURNISHING, PERFORMANCE, INSTALLATION, OR USE OF THE SOFTWARE PRODUCT, WHETHER DUE TO A BREACH OF CONTRACT, BREACH OF WARRANTY, OR THE NEGLIGENCE OF LICENSOR OR ANY OTHER PARTY, EVEN IF LICENSOR IS ADVISED BEFOREHAND OF THE POSSIBILITY OF SUCH DAMAGES.

TO THE EXTENT THAT THE APPLICABLE JURISDICTION LIMITS LICENSOR’S ABILITY TO DISCLAIM ANY IMPLIED WARRANTIES, THIS DISCLAIMER SHALL BE EFFECTIVE TO THE MAXIMUM EXTENT PERMITTED. IN NO EVENT SHALL LICENSOR’S TOTAL LIABILITY TO YOU FOR ALL DAMAGES EXCEED THE UNUSED PORTION OF THE AMOUNT PAID FOR THE LICENSE.

## Governing Law, Jurisdiction and Costs

This Agreement is governed by the laws of Texas, without regard to Texas’s conflict or choice of law provisions. If you are not a U.S. citizen or do not reside in the U.S., you hereby agree that any dispute or claim arising from this Agreement shall be governed by the applicable laws identified above and you herby irrevocably submit to the non-exclusive jurisdiction of the courts located in the jurisdiction identified above.

## Severability

If any provision of this Agreement shall be held to be invalid or unenforceable, the remainder of this Agreement shall remain in full force and effect. To the extent any express or implied restrictions are not permitted by applicable laws, these express or implied restrictions shall remain in force and effect to the maximum extent permitted by such applicable laws.

## Using Excel

Excel is a common tool used to view and edit .csv (comma-separated values) files. However, there are some things you need to consider when doing this.

- Excel may not properly handle the delimiter, and you may need to define the delimiter yourself
- Excel will pad all lines with multiple entries with enough commas to make the lines equal in length, this is especially apparent with the Criteria and Negative Criteria sections

## Installing Pairwise

The following steps are required in properly install AnkrPt Pairwise. Feel free to skip those already completed.

1. Install JDK 11 **or higher**
2. Download the jar file from Pairwise

Pairwise is ready to launch!

If you face troubles running GUI on macOS or Ubuntu install JavaFX manually.

For java JDK installation refer to:
installation for Windows,
installation for macOS.

## [Optional] Installing JavaFX on macOS:

1. Go to https://gluonhq.com/products/javafx/ and download JavaFX Mac OS X SDK.
2. Navigate to Downloads and unzip the file by double-clicking on it.
3. In your terminal, run the following command:

```
export PATH_TO_FX="[path-to-unziped-javafx]/javafx-sdk-[verion]/lib"
```

1. Run Pairwise in terminal with the following command:

```
java --module-path $PATH_TO_FX --add-modules javafx.controls,javafx.fxml,javafx.swing -jar pairwise-
1.00.RELEASE.jar
```

You will need to do export PATH_TO_FIX step each time you'll run terminal as variable does not carry over between terminal windows.

## [Optional] Installing JavaFX on Ubuntu (18.04 or newer):

1. Open a command line.
2. Type `apt install openjfx` to get the package. It is installed to /usr/share/openjfx/lib.
3. In terminal, run the following command:

```
export PATH_TO_FX="/usr/share/openjfx/lib"
```

1. Run Pairwise in terminal with the following command:

```
java --module-path $PATH_TO_FX --add-modules javafx.controls,javafx.fxml,javafx.swing -jar pairwise-
1.00.RELEASE.jar
```

You will need to do export PATH_TO_FIX step each time you'll run terminal as variable does not carry over between terminal windows.

# A Pairwise Example in Practice

## Challenge:

Testing part of application that helps people to set medical appointments. Apart from appointments in proper facility people may get advice by phone, home-visit or get general advice for certain group of symptoms.

## Task:

Creation of Test suite that will cover all paths for indicated system.

## Steps required to achieve the goal and conclusions:

1. Analyze input data.
   After deep analysis of possible input values, data set was limited to the values that provide various path coverages for tested application. Values were chosen to be the best representatives of data that interacts with each other, and to discover defects that may have critical impact on the tested system.
2. Provide data for AnkrPt `Pairwise` and combinatorial testing.

   We created input categories with indicated positive values:

   ```
   CRITERIA NAME   VALUES
   Person          child, adult, child under 2,
   MD              pediatrician, internist, dermatologist,
   Purpose         control check, prescription, infection,
   Symptoms        fever, rash & fever, rash, runny nose,
   Visit Type      phone, home-visit, emergency,
   ```

   **Assumptions:**

   For provided data we expect to have full coverage of the system. We can try testing all combinations of given values (getting `324` test cases = 3*3*4*3*3 - Cartesian Count) or try to use a shortcut to achieve it with Pairwise, making test case numbers significantly less.

   **Observations:**

   Pairwise by AnkrPt generates a Test Suite for this input data set containing `16` test cases, that are composed of 102 simple all-pairs of given values:

   | # | Person | MD | Purpose | Symptoms | Visit Type |
   |---|---|---|---|---|---|
   | 1 | child under 2 | pediatrician | prescription | fever | phone |
   | 2 | child | dermatologist | control check | runny nose | home-visit |
   | 3 | child | pediatrician | infection | rash & fever | phone |
   | 4 | adult | dermatologist | control check | runny nose | phone |
   | 5 | child under 2 | dermatologist | infection | rash | emergency |
   | 6 | child under 2 | internist | prescription | rash & fever | phone |
   | 7 | adult | pediatrician | control check | rash | phone |
   | 8 | adult | internist | infection | fever | home-visit |
   | 9 | child under 2 | internist | control check | rash & fever | home-visit |
   | 10 | child under 2 | pediatrician | prescription | runny nose | phone |
   | 11 | adult | pediatrician | control check | fever | emergency |
   | 12 | child | internist | prescription | rash | home-visit |
   | 13 | child | dermatologist | control check | fever | emergency |
   | 14 | child | pediatrician | infection | rash | home-visit |
   | 15 | adult | dermatologist | prescription | rash & fever | emergency |
   | 16 | adult | internist | infection | runny nose | emergency |

   **Conclusions:**

   We can see that some pairs of values appear more than once, but this is inevitable, as covering test cases requires return values exactly fulfilling indicated path (to cover minimum amount of test cases with all pairs indicates creating cases where some parameters can have any value from given positive set of data).

   Comparing testing all combinations of values to Pairwise numbers makes a huge difference between 324 and 16 test cases.
3. Provide data for AnkrPt `Pairwise` to mitigate risks connected with Pairwise technique.

   **Assumptions:**

It is known that pairwise technique fails when highly probable or high-impact input value combinations get too little attention. To mitigate those risks we use Required Tests, to assure that those combinations are included in generated Test Suite.

We define required tests for given example (empty values stand for "any valid value"):

Table 1. Test Suite containing Pairwise Test Cases

| ID | Person | MD | Purpose | Symptoms | Visit Type |
|---|---|---|---|---|---|
| 1 | | dermatologist | prescription | rash | phone |
| 2 | child | pediatrician | | | home-visit |
| 3 | child under 2 | | | rash & fever | emergency |

**Observations:**

Another run of Pairwise and we get brand new, polished Test Suite composed of 15 test cases, that includes high-risk / highly probable value combinations:

Table 2. Test Suite containing Required Tests as a Pairwise Test Cases

| # | Person | MD | Purpose | Symptoms | Visit Type |
|---|---|---|---|---|---|
| 1 | child under 2 | internist | control check | rash | emergency |
| 2 | adult | pediatrician | control check | rash & fever | home-visit |
| 3 | adult | dermatologist | infection | runny nose | home-visit |
| 4 | adult | internist | control check | rash | emergency |
| 5 | child under 2 | pediatrician | prescription | fever | home-visit |
| 6 | child under 2 | pediatrician | prescription | fever | phone |
| 7 | child | internist | prescription | rash & fever | phone |
| 8 | child | dermatologist | infection | rash | phone |
| 9 | adult | dermatologist | prescription | rash | phone |
| 10 | adult | internist | infection | fever | home-visit |
| 11 | child | pediatrician | infection | rash | home-visit |
| 12 | child | internist | control check | runny nose | phone |
| 13 | child under 2 | dermatologist | infection | rash & fever | emergency |
| 14 | child under 2 | pediatrician | prescription | runny nose | emergency |
| 15 | child | dermatologist | control check | fever | emergency |

**Conclusions:**

Adding required test to the suite does not mean that number of test cases will increase.

`Pairwise` processes them at the beginning of the test case optimization, so they have impact on the process. There are situations where number of test cases may be smaller with required test than without them and this is proper behavior, as `Pairwise` will give you the best test suite with given input, and it may happen that required test will be better fit for further test generation.

4. Don’t forget about negative testing!
**Assumptions:**

As we’ve got complete Test Suite covering "happy paths" for the application it is needed to provide tests covering negative values.

We provide negative values that should be included in Test Suite:

```
CRITERIA NAME    VALUES
Person           elder,
MD               nurse, surgeon,
Symptoms         none, pain,
Visit Type       chat, none,
Purpose          giving birth,
```

**#Observations:**

After another run of Pairwise we have completed Test suite composed with 15 "happy path" test cases (including required ones) and 8 negative test cases.

Table 3. Test Suite containing Required Tests

| # | Person | MD | Purpose | Symptoms | Visit Type | <INVALID> |
|---|---|---|---|---|---|---|
| 1 | child under 2 | internist | control check | rash | emergency | |
| 2 | child under 2 | internist | control check | rash | chat | Visit Type |
| 3 | adult | internist | control check | rash | emergency | |
| 4 | child under 2 | pediatrician | prescription | runny nose | emergency | |
| 5 | child | internist | prescription | rash & fever | phone | |

| # | Person | MD | Purpose | Symptoms | Visit Type | <INVALID> |
|---|---|---|---|---|---|---|
| 6 | adult | dermatologist | prescription | rash | phone | |
| 7 | adult | surgeon | prescription | runny nose | home-visit | MD |
| 8 | child under 2 | nurse | infection | rash & fever | emergency | MD |
| 9 | elder | pediatrician | control check | rash | phone | Person |
| 10 | child | dermatologist | control check | fever | emergency | |
| 11 | child | dermatologist | giving birth | fever | phone | Purpose |
| 12 | child | dermatologist | infection | rash | phone | |
| 13 | adult | pediatrician | control check | rash & fever | home-visit | |
| 14 | adult | dermatologist | infection | runny nose | home-visit | |
| 15 | adult | internist | infection | fever | home-visit | |
| 16 | child | internist | prescription | pain | home-visit | Symptoms |
| 17 | child | internist | control check | runny nose | phone | |
| 18 | child under 2 | dermatologist | infection | rash & fever | emergency | |
| 19 | adult | pediatrician | infection | rash & fever | none | Visit Type |
| 20 | child | pediatrician | infection | rash | home-visit | |
| 21 | adult | dermatologist | infection | none | emergency | Symptoms |
| 22 | child under 2 | pediatrician | prescription | fever | phone | |
| 23 | child under 2 | pediatrician | prescription | fever | home-visit | |

```
AnkrPt PairWise Processing  for "c:\tmp\test.csv"
-----------------------------
Columns         - 5
  Largest       - 4
  Smallest      - 3

Valid Values    - 16
Negative Values - 8
Total Pairs     - 102
Cartesian Count - 324

Total Tests     - 15
  Required      - 3    (*** included in total tests)
  Negative      - 8    (*** not included in total tests)

Test data results saved to 'c:\tmp\test-output.csv'
```

**Conclusions:**

AnkrPt Pairwise is giving you Test Suite containing 15 positive test cases and 8 negative, to cover all system paths, highly probable and high-risk value combinations to give you opportunity to discover most defects that might occur during complex combinatorial testing.

5. Improve your Test Suite with CONSTRAINTS!
We provide constraints for our Test Suite: We know that `child` and `child under 2` should be always treated by `pediatrician`, and `rash` as well as `rash and fever` should be treated by `dermatologist` or `pediatrician`(as it may be children issue).

| | | | |
|---|---|---|---|
| When | Person | isIn | child,child under 2 |
| Then | MD | is | pediatrician |
| When | Symptoms | isIn | rash & fever,rash |
| Then | MD | isIn | dermatologist,pediatrician |

During test generation, we get information, about pairs that were eliminated with Constraints:

```
AnkrPt PairWise Processing  for "c:\tmp\test.csv"
-----------------------------
Columns         - 5
  Largest       - 4
  Smallest      - 3

Valid Values    - 16
Negative Values - 8
Total Pairs     - 96
Cartesian Count - 324

Total Tests     - 17
```

```
  Required      - 3    (*** included in total tests)
  Negative      - 8    (*** not included in total tests)
Generated in    - 55 ms

Pairs filtered by constraints:
0-child <> 1-dermatologist
0-child <> 1-internist
0-child under 2 <> 1-dermatologist
0-child under 2 <> 1-internist
3-rash & fever <> 1-internist
3-rash <> 1-internist

Test data results saved to 'c:\tmp\test-output.csv'
```

ⓘ Remember that numbers before criteria values stands for criteria index (order criteria were defined in `Criteria` section) to make it easier to identify them.

| # | Person | MD | Purpose | Symptoms | Visit Type | <INVALID> |
|---|---|---|---|---|---|---|
| 1 | adult | internist | prescription | fever | home-visit | |
| 2 | adult | dermatologist | control check | fever | chat | Visit Type |
| 3 | adult | nurse | control check | fever | emergency | MD |
| 4 | child under 2 | pediatrician | prescription | rash & fever | emergency | |
| 5 | adult | dermatologist | prescription | rash | phone | |
| 6 | child under 2 | pediatrician | control check | rash | home-visit | |
| 7 | child under 2 | pediatrician | infection | fever | phone | |
| 8 | child | pediatrician | control check | rash & fever | home-visit | |
| 9 | adult | dermatologist | giving birth | fever | emergency | Purpose |
| 10 | child | pediatrician | infection | rash | emergency | |
| 11 | adult | surgeon | control check | fever | emergency | MD |
| 12 | adult | internist | control check | runny nose | emergency | |
| 13 | adult | internist | infection | runny nose | phone | |
| 14 | child | pediatrician | control check | rash & fever | phone | |
| 15 | adult | pediatrician | control check | rash | emergency | |
| 16 | adult | dermatologist | control check | none | emergency | Symptoms |
| 17 | adult | dermatologist | prescription | runny nose | home-visit | |
| 18 | adult | dermatologist | control check | fever | none | Visit Type |
| 19 | child | pediatrician | prescription | fever | phone | |
| 20 | child under 2 | pediatrician | control check | runny nose | emergency | |
| 21 | adult | dermatologist | control check | fever | emergency | |
| 22 | child | pediatrician | infection | runny nose | home-visit | |
| 23 | adult | dermatologist | control check | pain | emergency | Symptoms |
| 24 | elder | dermatologist | control check | fever | emergency | Person |
| 25 | adult | dermatologist | infection | rash & fever | phone | |

**Conclusions:**

After another run of Pairwise we have completed Test suite composed with 17 "happy path" test cases (including required ones and constraints) and 8 negative ones, but those are more suited for our needs(as we can see, required test changed to meet constraints conditions).

6. CONCLUSIONS:
Summing up, AnkrPt `Pairwise` is a very handy tool that will make your work much easier. Even with complex Test Suite composed of positive, negative criteria, required test and constrains you will get significantly less test cases than with all value combinations (17 "happy path" cases and 8 negative ones, with constraints to 324 Cartesian Count that do not even cover negative cases).

Pairwise require some time in preparing the data, constraints and requirements, as well as understanding the business processes. Testing planing with AnkrPt `Pairwise` is less time and budget consuming, easier to estimate, and easier to manage with application updates (once you have created your initial set, you can easily update as changes occur).

## Example Run Scenarios

All examples assume the jar is at "**c:\pairwise\pairwise.jar**" and your source folder at "**c:\tests\pairwise**". Examples also assume the output file already exists and identifies the scenario when renaming occurs.

### Example from source without output

This example assumes you are currently in the source folder at **c:\tests\pairwise\**. If the output file exists during generation, "**_output-99**" will be added to the output filename

✓ java -jar c:\pairwise\pairwise.jar abc.csv

- reads from **c:\tests\pairwise\abc.csv**
- saves to **c:\tests\pairwise\abc-output.csv**.
- **or** saves to **c:\tests\pairwise\abc-output.csv** after renaming existing to **c:\tmp\pairwise\abc-output (copy 12).csv**

### Example from source with relative folder

This example assumes you are currently in the source folder at **c:\tests\pairwise\**.

✓ java -jar c:\pairwise\pairwise.jar abc.csv **output**

- reads from **c:\tmp\pairwise\abc.csv**
- saves to **c:\tmp\pairwise\output\abc.csv**
- **or** saves to **c:\tests\pairwise\abc.csv** after renaming existing to **c:\tmp\pairwise\abc (copy 12).csv**

### Example from source with explicit file

This example assumes you are currently in the source folder at **c:\tests\pairwise\**.

✓ java -jar c:\pairwise\pairwise.jar c:\work\abc.csv output\output.csv

- reads from **c:\tests\abc.csv**
- saves to **c:\tests\output\output.csv** and ***overwrites*** if it already exists

### Example from runtime with all options

✓ java -jar pairwise.jar c:\tests\pairwise\abc.csv output -delimiter="|" -charset="ISO-8559-1"

- reads from **c:\tests\pairwise\abc.csv** which had to be absolute pathed
- saves to **c:\tests\pairwise\output\abc.csv**
- if it already exists, saves to **c:\tests\pairwise\output\abc (copy 1).csv**

# Configuration Files

AnkrPt Pairwise uses simple **csv** files to define how tests generate. These files contain up to four (4) different sections as described below.

## Sections

| Section | Required | Description |
|---|---|---|
| **CRITERIA** | **yes** | identifies the criteria used to generate tests |
| **NEGATIVE CRITERIA** | no | defines negative criteria used to generate negative tests |
| **REQUIRED TESTS** | no | identifies specific tests to generate. All fields need not be included. |
| **CONSTRAINTS** | no | define limitations to be enforced during test generation |

## Basic Rules

There are some basic rules when editing your csv (comma-separated values) files:

- section keywords are uppercase and define the beginning of each section,
- each section can only occur once,
- section order is vital as each section builds upon previous ones,
- only the **CRITERIA** section is required; all others are optional,
- all values must be separated by a delimiter (regional defaults),
- if you don’t want the regional delimiter, a runtime parameter lets you override it,
- you can edit with Excel but **must retain the csv structure**; we do not support native Excel,
- blank lines are ignored so can be used for readability.

## Criteria

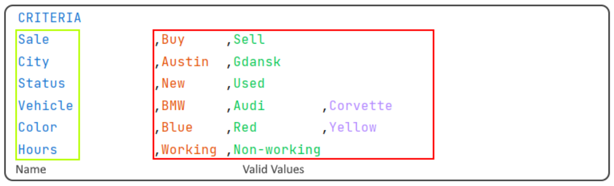
The Criteria section defines the valid data used to generate tests. The structure of the Criteria section is:

```
CRITERIA <b class="conum">(1)</b>
Name, value1, value 2, "value3", ... <b class="conum">(2)</b>
```

1. CRITERIA tag identifying the start of the Criteria section.

2. a criteria entry with the criteria name followed by a list of valid criteria. *The name is in blue below.*

Let’s see an example of a CRITERIA section with valid criteria.



```
CRITERIA
Sale        ,Buy     ,Sell
City        ,Austin  ,Gdansk
Status      ,New     ,Used
Vehicle     ,BMW     ,Audi       ,Corvette
Color       ,Blue    ,Red        ,Yellow
Hours       ,Working ,Non-working
```

CRITERIA section example

*Figure 1. CRITERIA section example*

*Please refer to Using Excel for additional considerations for working with csv files in Excel.*

This structure makes it easy to easily define your requirements while still allowing complex testing conditions. Some basic rules for Criteria are:

- The Criteria section starts with the **CRITERIA** tag.
- There must be one, and only one, Criteria section.
- Each criteria row defines the criteria name *(in blue above)* and corresponding valid values for that criteria.
- At least two criteria are required.
- Each criterion can only be defined once. If criteria are defined multiple times, you must merge them.
- Each criterion must have at least two values with all values unique for a given criteria.
- All names and values are automatically trimmed (removal of leading and trailing spaces).
- To use the delimiter within values, use quotes as per normal csv processing.
- Blank lines are ignored.

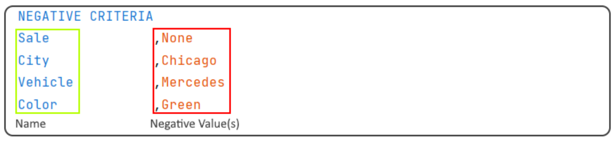
## Negative Criteria

It is not enough to just do 'happy path' testing, you should do negative testing as well to ensure the system is stable even with harmful parameters. AnkrPt Pairwise does this with **NEGATIVE CRITERIA** as follows:

```
NEGATIVE CRITERIA <b class="conum">(1)</b>
Name, value1, value 2, "value3", ... <b class="conum">(2)</b>
```

1. NEGATIVE CRITERIA tag identifying the start of the Negative Criteria section
2. each row identifies the name of a valid criterion followed by a list of invalid values

Let’s see an example of a NEGATIVE CRITERIA section with **invalid** criteria.



```
NEGATIVE CRITERIA
Sale        ,None
City        ,Chicago
Vehicle     ,Mercedes
Color       ,Green
```

Negative Criteria Example

*Figure 2. Negative Criteria Example*

It is similar to the CRITERIA section except for negative values, and you must refer to an existing criteria **name** from the CRITERIA section. Some basic rules for Negative Criteria are:

- The Negative Criteria section starts with the **NEGATIVE CRITERIA** tag.
- There must be one, and only one, Negative Criteria section.

- Each negative criteria defines a criteria name and corresponding negative values.
- Each criterion must reference an existing valid criteria from the CRITERIA section.
- You cannot define a negative value if it is already a valid value for that name.
- Each negative criteria are optional.
- All names and values are automatically trimmed (removal of leading and trailing spaces).
- To use the delimiter within values, use quotes as per normal csv processing.
- Blank lines are ignored.

By default, Pairwise generates as many negative test cases as there are negative values (one invalid value per Test Case). You can change this to condense multiple negative tests into as few tests possible producing tests with multiple negative conditions on a single row.

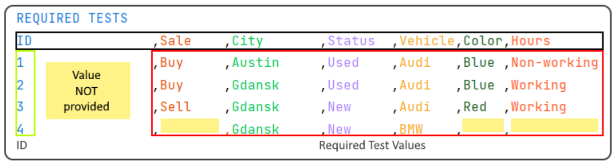
## Required Tests

Generating pairwise tests ensures all pairs are tested, but not necessarily a specific test that you need. Required tests are a quick and easy to ensure a specific test case is produced. Let’s look at the structure.

```
REQUIRED TEST <b class="conum">(1)</b>
ID, Name1, Name2, Name3, Name4, Name5,... <b class="conum">(2)</b>
1,value1,,,,value5,... <b class="conum">(3)</b>
```

1. **REQUIRED TEST** tag identifies the start of the Required Test section.
2. A label line making it easier to match your fields to the required value.
   The first column is a "Test ID" column.
3. The id of the required test followed by values for each name.

Let’s see  an example of the REQUIRED TEST section.



```
REQUIRED TESTS
ID                  ,Sale    ,City         ,Status    ,Vehicle,Color,Hours
1                   ,Buy     ,Austin       ,Used      ,Audi    ,Blue ,Non-working
2                   ,Buy     ,Gdansk       ,Used      ,Audi    ,Blue ,Working
3                   ,Sell    ,Gdansk       ,New       ,Audi    ,Red  ,Working
4                   ,        ,Gdansk       ,New       ,BMW     ,     ,
```

Required tests example

*Figure 3. Required tests example*

Required tests are a little different from the Criteria sections but refer to those sections for data. Some basic rules for Required Tests are:

- refer to values defined in either the CRITERIA or NEGATIVE CRITERIA sections.
- undefined criteria are filled with criteria values during generation.
- at least three criteria are required otherwise you’re just doing pair generation which is already done.

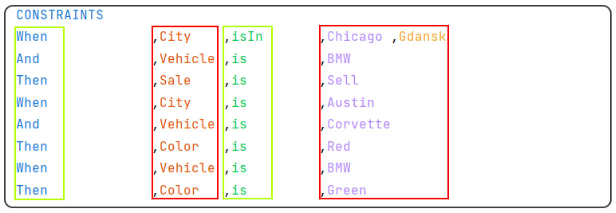
## Constraints

Constraints are an extremely important and powerful part of AnkrPt Pairwise. They let us define limits while building test cases, making it possible to ensure proper tests. Constraints use the following structure:

```
CONSTRAINTS <b class="conum">(1)</b>
When [criteria] is/isIn/isNot/isNotIn [value] <b class="conum">(2)</b>
And/Or [criteria] is/isIn/isNot/isNotIn [value] <b class="conum">(3)</b>
Then [criteria] is/isIn/isNot/isNotIn [value] <b class="conum">(4)</b>
And [criteria] is/isIn/isNot/isNotIn [value] <b class="conum">(5)</b>
```

1. **CONSTRAINTS** tag identifying the start of a Constraints section.
2. **When** defines the beginning of a constraint
3. **And** and **Or** define additional criteria for the constraint
4. **Then** is the beginning of the **Then** section for its associated **When**
5. **And** defines additional **Then** criteria. *Or is not allowed with Then.*

Let’s see an example.



```
CONSTRAINTS
When              ,City    ,isIn        ,Chicago ,Gdansk
And               ,Vehicle ,is          ,BMW
Then              ,Sale    ,is          ,Sell
When              ,City    ,is          ,Austin
And               ,Vehicle ,is          ,Corvette
Then              ,Color   ,is          ,Red
When              ,Vehicle ,is          ,BMW
Then              ,Color   ,is          ,Green
```

Constraints example

*Figure 4. Constraints example*

Some basic rules of the Constraints section are:

- every constraint requires both a **When** and **Then** statement.
- **Or** is not allowed with the **Then** clause.
- **And** and **Or** follow basic programming logic with **Or** tested first followed by **And**.

Constraints are built with keywords, criteria, and values (including negative ones) based upon basic logical conditions using a **"When *this* Then *that*"** structure.

## Keywords

Table 1. Constraints keywords

| **Keyword** | **Description** |
|---|---|
| When | defines the beginning of a Constraint |
| Then | defines the beginning of the **Then** clause |
| And | logic operator to **And** your criteria |
| Or | logic operator to **Or** your criteria |
| Is | lets you define a **single** value that the condition **must match** |
| IsIn | defines **multiple** possible values that the Constraint must **match one** of |
| IsNot | defines a single value which the Constraint **cannot be** |
| IsNotIn | defines a list of values which the Constraint **cannot match any** of |

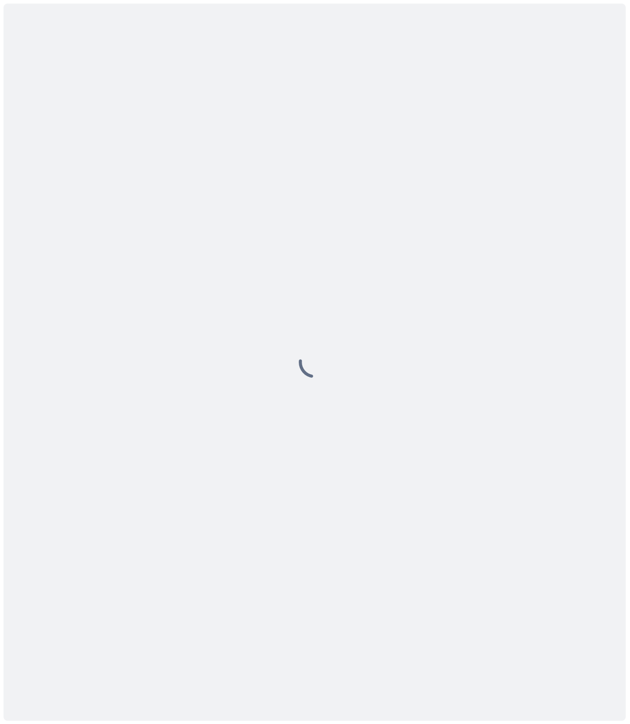
- **THEN** conditions should be based on explicit requirements.
- **WHEN** and **THEN** are defined once per Constraint; usage of both is mandatory.
- you must use defined criteria.
- **REQUIRED TESTS** are less important than **CONSTRAINTS** - it is required to resolve conflict before test generation.
- **OR** is not allowed to define multiple values for the same criteria, use **isIn** or **isNotIn** instead.

It’s essential to use **CONSTRAINTS** wisely otherwise you might accidentally eliminate some needed pairs from test generation. You should verify the dependencies between constraints to avoid the risk of omitting essential pair combinations.

To better understand the impact of your constraints, pairs not considered valid for tests is provided in the generation summary.

## Putting it All Together

Now let’s put everything together to see a complete sample Criteria csv file.



Putting it all together

*Figure 5. Putting it all together*

Download example as a CSV file

## Messages

The following messages can be generated during AnkrPt Pairwise generation.

| Code | Message | Description |
|---|---|---|
| 101 | Cannot add additional criteria once required tests are defined. | To help manage data, we enforce all positive criteria created first, followed by negative criteria. Then required tests, and finally constraints. Forcing this order simplifies effectively managing your test criteria. |
| 102 | Section <> cannot be defined more than once, please combine the two sections. | You can only define each section once in the following order: Criteria, Negative Criteria, Required Tests, Constraints. |
| 103 | Criteria <> is already defined and cannot be defined again. | You've already defined values for the identified criteria. Make sure to provide all the valid values together, and all the negative values together. |
| 104 | <> has not yet been defined in your criteria. | You can't refer to a criteria that has not yet been defined. Check to see that you've spelled the criteria name correctly. |
| 105 | <> needs at least two criteria defined. | If you're doing pairwise testing, we need to generate pairs, meaning two. You need at least two criteria to test. |
| 106 | <> cannot be added multiple times to <> | Don't add the same value multiple times. Duplicated data causes duplicated tests and the purpose of pairwise testing is to maximize coverage while minimizing testing. Extra blanks are considered extraneous, so we check for this but don't automatically remove them; you must manage your own test data. |
| 107 | 'CRITERIA' line missing. | Every Criteria file must start with CRITERIA as the first line otherwise it is not considered a Criteria definition file. This lets us support BOM markers. |
| 108 | Criteria not defined. | Pairwise needs to have criteria defined in order to produce tests. |
| 109 | All the valid and negative values for each criterion must be defined together. | You must group all the valid or negative values for each criterion at the same time, you cannot add additional valid or negative for a criteria after already having defined them. |
| 110 | Criteria name must be provided, blank is not allowed. | |
| 201 | <> is already defined as valid criteria for <>. | You have already defined the identified criteria as valid, so trying to include it for negative testing is a conflict. Make it either positive or negative, not both. |

| | | |
|---|---|---|
| 301 | Required tests must include valid criteria. <> cannot be <>. | Required tests are a quick and easy way to ensure specific happy path tests. If you want to test more complex processing, this must be done with Constraints. |
| 302 | The row immediately after REQUIRED TESTS must be a header row matching the criteria name <>. | Required Tests expect a header row of the criteria names to ensure you're entering your Required Tests in a consistent manner, the order defined in CRITERIA. But don't forget the ID column at the beginning which simplifies Required Test management, and allows identification of negative Required Tests. |
| 303 | Required Test <> requires at least 3 criteria. | All Required Tests must include at least three (3) criteria otherwise they're just pairwise tests. And all pairwise tests are automatically generated. |
| 304 | Too many criteria (<>) provided for Required Test."<>". | You tried passing more criteria to your Required Test than are defined in CRITERIA. |
| 305 | Required test '<>' conflicts with '<>'. | Both Required Tests test the same thing so you must choose which to use. This often occurs when one Required Test has fewer criteria than another but the criteria match. You likely want the more explicit test but you have to make that decision. |
| 306 | Required test ID must be unique. '<>' already defined. | The id of the Required Test must be unique otherwise we can't identify which Required Tests must be reviewed when there are issues. |
| 307 | Required test ID must be provided, blank is not allowed. | Each Required Test must provide a unique id to identify Required Tests must be reviewed when there are issues. |
| 308 | Required Test #<> is not valid. | Each Required Test must be checked by provided constraints. If it is not valid, then it cannot be added. |
| 401 | Constraint #<> - Or is not allowed on THEN clauses. Use isIn or isNotIn if necessary. | THEN is expecting a very specific condition across criteria so OR is not allowed across criteria. However, if you want OR for a single criteria, isIn and isNotIn will support this. |
| 402 | Constraint #<> - WHEN can only be defined once. | You can only define WHEN once. Did you mean to use AND or OR instead? |
| 403 | Constraint #<> - THEN can only be defined once. | THEN can only be defined once for a Constraint. Did you mean to use AND instead? |
| 407 | Constraint #<> - Cannot have a THEN without a WHEN | THEN doesn't logically work without a WHEN. Add a WHEN to define your Constraint. |
| 408 | Constraint #<> - WHEN must be defined before <> | Constraints must be entered as WHEN, AND/OR, THEN, AND. |

| | | |
|---|---|---|
| 409 | Only one value is allowed when using <> | IS and ISNOT only allow a single value. ISIN and ISNOTIN allowed for multiple values. |
| 410 | More than one value is required when using <> | ISIN and ISNOTIN expect multiple values. Use IS and ISNOT if you're comparing against only one value. |
| 411 | Merge your multiple OR statements for criteria <> into a single ISIN/ISNOTIN statement. | Merging multiple OR statements for the same criteria into a single statements makes it is easier to understand and manage the conditions of your criteria. |
| 412 | The provided constraint condition (<>) is invalid for line "<>". | |
| 413 | Row (<>) cannot be filled with values because of opposed constraints. | |
| 414 | One of the values provided for constraint #<> is empty. | Check if your constraint lines doesn't have any unnecessary delimiters at the end. |
| 415 | Criterion <> is repeated in <> for constraint #<>. | |
| 416 | Criterion <> exists in both WHEN and THEN part of constraint #<> | |
| 418 | Keyword is wrong or missing for line <> | Possible keyword are: When, Then, Or, And |
| 501 | Error reading file. <> | The system failed reading the file identified. Ensure the file exists at the location identified. |
| 502 | Error writing file "<>". Please ensure file is not open, locked or doesn't have invalid characters in the filename. | The system failed trying to open or write to the file identified. Ensure you have access to the file location, and that the file is not currently open, locked by another user or process or doesn't have invalid characters in filename. |
| 503 | File "<>" is using an unsupported file type "<>". Please contact Kimputing. | The file type "<>" is unsupported. AnkrPt Pairwise currently only supports 'csv' files. Contact Kimputing is additional file types are needed. |
| 504 | The file <> needs a file extension to be processed. | The system needs to know the filetype being processed and uses the file extension to determine this. Please ensure the file has a file extension. |
| 505 | <> is an unsupported encoding type. | The provided encoding is not supported by Java. Please try something different. |
| 506 | The input and output filenames must be distinct. | Providing the same input and output filenames will result in your input file being overridden. This is not allowed. |
| 507 | Error renaming file | *additional information provided* |
| 508 | An input filename must be provided." | |

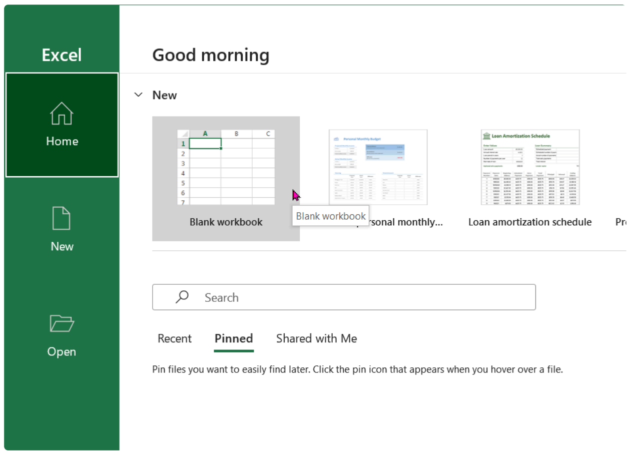
| | | |
|---|---|---|
| 510 | The charset "<>" provided is not valid. | The character set provided is not a valid one. Choose another such as UTF-8 or ISO-8859-1. |
| 511 | The additional output "<>" provided is not valid. | Valid additional output types are "none", "json", or "xml". |
| 512 | Invalid filename: <> | The provided filename is invalid, please correct it. |
| 601 | Invalid number of parameters. Parameter 1: input file. Parameter 2 (optional): output file. File locations can include paths. | |
| 602 | Criteria is too complex to generate sustainable tests. Reconsider the following and try again. <>. | Refer to the AnkrPt Pairwise Testing Data to understand how to leverage test data. If you need to override, use -override=yes but this is unsupported. |
| 699 | Pairwise test generation cancelled. | |
| 901 | Pairwise license key issue, please contact Kimputing at kimputing.com. | |
| 902 | Invalid or expired Pairwise license key, please contact Kimputing at kimputing.com. | |
| 903 | Pairwise license not found. | |
| 904 | License file not found. | Please review documentation for the location of the license file. |
| 905 | ANKRPT_PROPERTIES is deprecated. Please replace it with ANKRPT_USERFOLDER which points to AnkrPt user folder. | *additional information provided* |

## Resolving Input File Issues

Troubles with input files may occur sometimes. Pairwise allows to use files with custom delimiters, and charsets(java recognizable) - it is important to check if rules for custom setup are indicated (Running Pairwise).

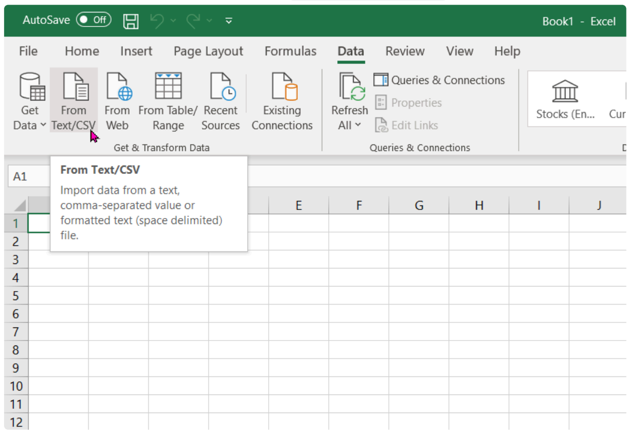
It is infrequent, but some csv files need to be rewritten to be valid for Pairwise. The easiest way to achieve this goal is to process it with Excel.

- Open a blank workbook.



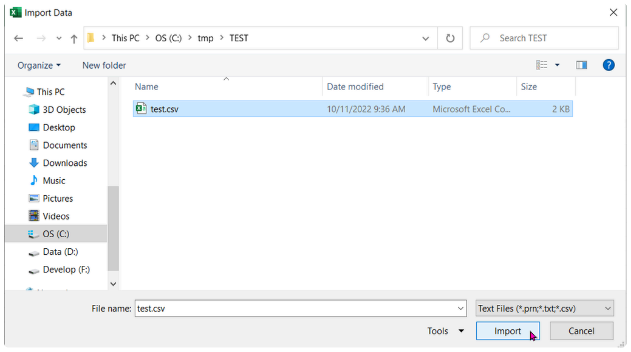
Pin files you want to easily find later. Click the pin icon that appears when you hover over a file.

- Change tab to DATA, click on `From Text/CSV`.



Import data from a text, comma-separated value or formatted text (space delimited) file.

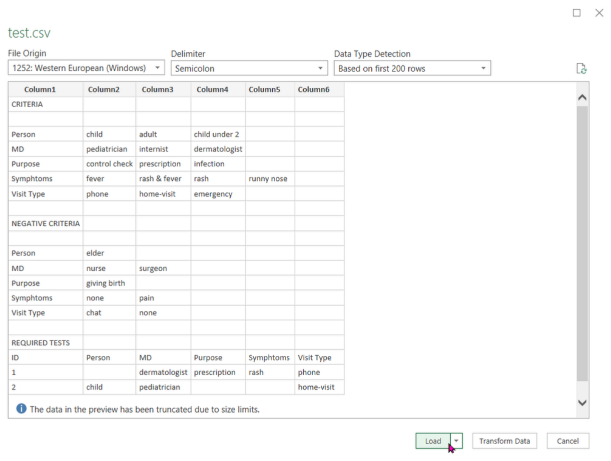
- Choose the not-working file, and click `Import`.



| Name | Date modified | Type | Size |
|---|---|---|---|
| test.csv | 10/11/2022 9:36 AM | Microsoft Excel Co... | 2 KB |

File name: test.csv

- Load data from CSV; click `Load`.



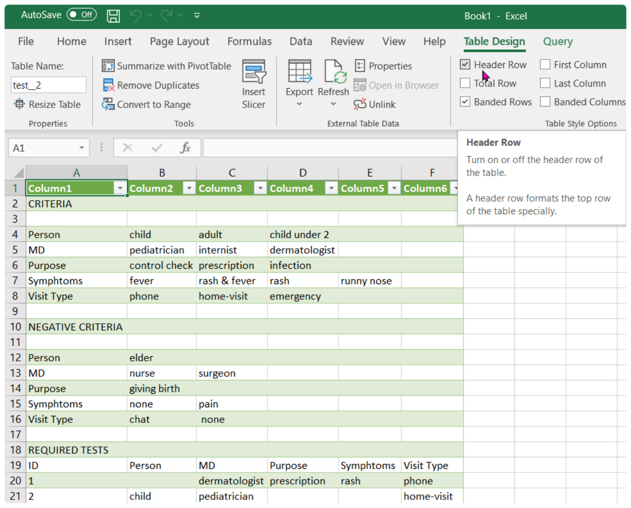
test.csv

File Origin: 1252: Western European (Windows)
Delimiter: Semicolon
Data Type Detection: Based on first 200 rows

| Column1 | Column2 | Column3 | Column4 | Column5 | Column6 |
|---|---|---|---|---|---|
| CRITERIA | | | | | |
| | | | | | |
| Person | child | adult | child under 2 | | |
| MD | pediatrician | internist | dermatologist | | |
| Purpose | control check | prescription | infection | | |
| Symphtoms | fever | rash & fever | rash | runny nose | |
| Visit Type | phone | home-visit | emergency | | |
| | | | | | |
| NEGATIVE CRITERIA | | | | | |
| | | | | | |
| Person | elder | | | | |
| MD | nurse | surgeon | | | |
| Purpose | giving birth | | | | |
| Symphtoms | none | pain | | | |
| Visit Type | chat | none | | | |
| | | | | | |
| REQUIRED TESTS | | | | | |
| ID | Person | MD | Purpose | Symphtoms | Visit Type |
| 1 | | dermatologist | prescription | rash | phone |
| 2 | child | pediatrician | | | home-visit |

The data in the preview has been truncated due to size limits.

- Remove header row from the file, uncheck `Header Row`



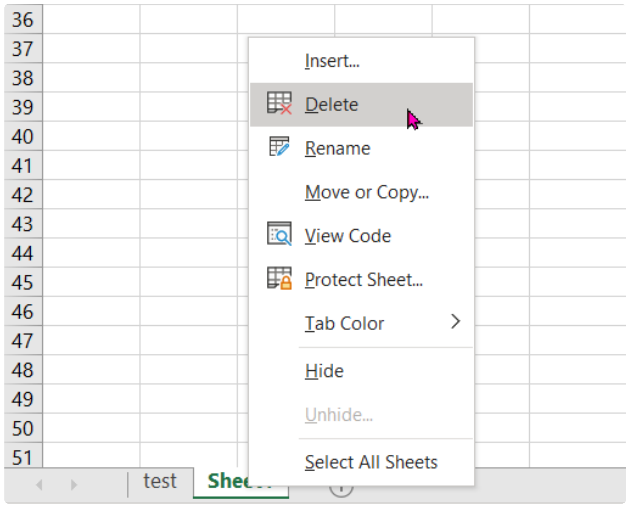
Header Row

Turn on or off the header row of the table.

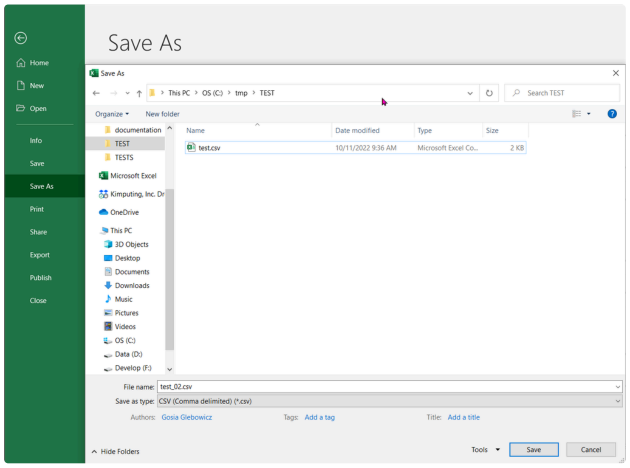
A header row formats the top row of the table specially.

| | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| 1 | Column1 | Column2 | Column3 | Column4 | Column5 | Column6 |
| 2 | CRITERIA | | | | | |
| 3 | | | | | | |
| 4 | Person | child | adult | child under 2 | | |
| 5 | MD | pediatrician | internist | dermatologist | | |
| 6 | Purpose | control check | prescription | infection | | |
| 7 | Symphtoms | fever | rash & fever | rash | runny nose | |
| 8 | Visit Type | phone | home-visit | emergency | | |
| 9 | | | | | | |
| 10 | NEGATIVE CRITERIA | | | | | |
| 11 | | | | | | |
| 12 | Person | elder | | | | |
| 13 | MD | nurse | surgeon | | | |
| 14 | Purpose | giving birth | | | | |
| 15 | Symphtoms | none | pain | | | |
| 16 | Visit Type | chat | none | | | |
| 17 | | | | | | |
| 18 | REQUIRED TESTS | | | | | |
| 19 | ID | Person | MD | Purpose | Symphtoms | Visit Type |
| 20 | 1 | | dermatologist | prescription | rash | phone |
| 21 | 2 | child | pediatrician | | | home-visit |

- Delete empty sheets( `csv` don’t support multiple sheets)



- Save file as `CSV (Comma delimited)(*.csv)`



This PC > OS (C:) > tmp > TEST

| Name | Date modified | Type | Size |
|---|---|---|---|
| test.csv | 10/11/2022 9:36 AM | Microsoft Excel Co... | 2 KB |

File name: test_02.csv

Save as type: CSV (Comma delimited) (*.csv)

Authors: Gosia Glebowicz Tags: Add a tag Title: Add a title

⚠ The default delimiter for EXCEL is depended on locale (regional settings) - it may be comma, semicolon, or other, which is set in operating system settings. To discover where to find information on how to deal with custom delimiters and charset, refer to Running Pairwise.

## Effective Test Data Selection

While it is easy to define criteria and generate tests, remember that pairwise is a way to maximize coverage while **minimizing** tests. It is crucial to **carefully** select criteria which adequately covers all desired combinations of values while minimizing the generated tests. Using values that don't change test coverage of the system may unnecessarily increase the number of unnecessary cases.

### Example 1

For example, 70 values distributed across parameters in two different ways can produce different results:

- 10 params with 7 values each (10*7) generates **86** tests (Cartesian > 282 Million)
- 7 params with 10 values each (7*10) generates **146** tests (Cartesian of 10 Million)

### More doesn't always mean more

Though it seems counterintuitive, adding more parameters or more values does not always mean more tests. Unfortunately, there is no formula to determine how many tests will be generated from criteria, the only way to know is to generate. But, you can know the fewest number of tests that will be generated; just multiply the size of the two largest columns together.

### Example 2

For example, 7 parameters in one set, and those same 7 criteria with an additional 25 criteria.

- 2 parameters of 10 values each and 5 parameters of 2 values each generates **100** tests (Cartesian of 3,200)
- those same 7 parameters **plus** 25 more of 2 values each ***still*** generates **100** tests (Cartesian > 3 Billion)

What does this all mean?

*That revealing the highest percentage of defects at minimum cost (minimum test cases with maximum test coverage) requires careful analysis of the values' scope.*

Using values that don't change test coverage of the system under test significantly increases the number of useless cases. The better you understand the data, the easier to define parameters that ensure all crucial test coverage parameters are defined. And that you aren't generating excessive and unnecessary tests.

It is important to remember that data selection is crucial to minimize testing.