Forked d2-C Test of Attention: Manual

Henrik Laxhuber

July 2017

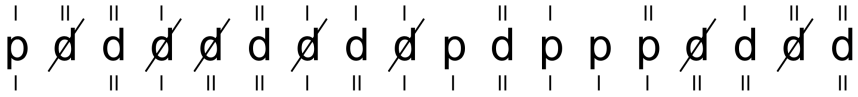
Contents

1 Introduction

1.1 The d2-Test

The d2-Test was originally devised by Rolf Brickenkamp in 1962 as an analogue test on paper. In 1997, Brickenkamp expanded this test to the computer with some colleagues; this test is now known as the d2-c test. This provides arguably the best stress test of attention to date. It consists of a row of symbols with dots above an below, as illustrated by Figure 1. The participant must cross out only all symbols 'd' with precisely two dots in total above and below the letter. In total, there are 12 sets of rows, with each set containing 7 rows. The time given to complete one set is 30 seconds. It is virtually impossible to complete all 7 rows in this time, the goal is thus to complete as many as possible. After these 30 seconds, the next set of rows

Figure 1: A d2 row as generated this implementation of the test.



commences, and the timer restarts. This process repeats until all 12 sets have been processed. In total, this leaves the pure test time at precisely 6 minutes.

1.2 Why recreate an existing test?

The d2-c test as created by Brickenkamp is part of the 'Hogrefe Testing System', which is a rather expensive software suite. However, the concept of the test is not protected by any patent laws or the like, wherefore the test can easily be reproduced. I decided to provide my implementation of the test to the public for free.

1.3 Functionality

The test hereby provided has an extensive functionality:

1. All data is collected centrally at a server.
2. Only the test administrator needs to install any software; the participants may access the test in the form of a web page.
3. Loading the test website for the first time displays an administrator page that allows to set treatment variables. These variables will be stored on the computer until the page refreshes.
4. Each participant is provided in-depth and easily comprehensible instructions on how to perform the test.
5. Each participant must complete a practice round before commencing with the test. An algorithm determines whether the participant has not understood the instructions, in which case the test administrator is sent a notification to their phone.
6. The symbol rows are generated randomly. The amount of sets of rows and the time for each such set can be modified.
7. The test administrator has a live preview of all collected data.
8. If a participant decides to leave the page, the administrator is sent a warning to their phone.

1.4 Measured Variables

The following list details all variables that can be measured with my computerised implementation of the d2 test. This test has been built largely according to the specifications by Brickenkamp, with some minor alterations:

TN The total number of symbols crossed out. This measure indicates the speed at which a participant worked.

FL The fluctuation as the maximum difference between the total number of items processed on the individual series.

$\mathbf{E_1}$ Errors of omission. These are the most common errors.

$\mathbf{E_2}$ Errors of confusion, split into the following:

- **$\mathbf{E_{2_D}}$** Incorrect number of dots. These are somewhat uncommon.
- **$\mathbf{E_{2_L}}$** Incorrect letter. These are least common.

E The total number of errors ($E_1 + E_2$).

E% The error percentage.

TN - E The total number of correctly recognised symbols.

CP The concentration performance score derived from $TN - E - E_2$.

$\mathbf{t_f}$ The time to complete each fragment resp. sub-row of a row. This value is outside the specifications by Brickenkamp; any conclusions derived from it must be treated with caution.

All data is stored per fragment or per sub-row, allowing to calculate the performance degradation over time using the t_f value. Again, these values are not specified by Brickenkamp, and have been implemented upon the rationale that the original paper implementation takes similar measurements.

I also decided to display twice as many symbols per sub-row as the reference specifies. This was due to modern technology having progressed considerably, providing larger screens of drastically higher resolution than what Brickenkamp worked with. However, this decision does not impact any measurements, for each sub-row is intrinsically divided into two pieces to conform to the reference specifications.

2 Setting Up

Setting up the test is somewhat complicated if you are not familiar with computers. This section aims to be as simple as possible; however, you will likely have to take multiple attempts to get a working test.

2.1 Prerequisites

The test administrator must install the necessary software to run the test server. This server runs on node.js, which can be obtained from `nodejs.org/`.

Once you have node.js installed, you must obtain a copy of the test software. If you have git, this is a simple matter of `git clone https://www.github.com/HenrikLaxhuber/d2test.git`, which you have likely already done. If you do not have git, you can download the necessary files by navigating to `github.com/HenrikLaxhuber/d2test` in your browser and clicking the 'Clone or Download' button. Extract the downloaded zip-archive to a new folder.

From a terminal window in the folder where the newly downloaded files are located, type `npm install`. This will take some time as it downloads roughly 50MB of data to your computer.

2.2 Deploying

Next, you must deploy the test server with your required configurations. To do so, type `npm run deploy` from within a terminal window. You will be asked which treatment groups your experiment requires. Type the name of each group without spaces, seperating the individual entries with a comma. For example, if you were testing whether standing yields a higher concentration performance than sitting, you would type `standing,sitting`. If you want to change your treatment groups, you will have to re-deploy.

For your phone to be able to receive notifications and status updates form the server, you must install the 'Pushbullet' app. Once you have done so, sign in on `pushbullet.com/` and find your access token in the settings. Then paste this access token into the prompt in your terminal window. Hit enter and then select the devices that you wish the server to push notifications to by typing either y(es) or n(o).

2.3 Starting the Server

To start the server, type `npm start`. By default, the server will run on port 33333, wherefore you must add a redirection to port 80 in your pre-routing tables. If all that means but garbage to you, you can attempt an easier solution: make sure that Skype is fully closed by right-clicking on the icon and selecting 'Quit'. Next, type `npm start 80`.

2.4 Accessing the Server

When you start the server, the program will display a link to access your test website. Type this link in a modern web browser on any machine in the same shared internet connection to access the test.

Do not close the terminal window in which the server is running. This will cause the server to terminate.

2.5 Closing the Server

To end the server, simply hit '[Ctrl] + C'.

3 Reading the Data

The data is collected centrally at the server and stored in an sqlite3 database. To view its contents, you will need to download the sqlite3 tools binaries for your system from `sqlite.org/download.html`. Copy the `sqlite3` executable to the folder containing the web server. Open a terminal window in this folder and run `sqlite3 app/data.db`. It is advisable to first create a backup of the database in case you accidentally delete the data.

The database contains two tables, 'received' and 'processed'. The former holds the raw data as received from each client; the latter holds processed data such as the variables described in subsection 1.4.

To learn more about the columns in each table, type `.schema`.

To view the values of specific columns, type `SELECT <column names> FROM <table>`, where the column names are comma-separated. If you want to view only the data where certain conditions apply, type `SELECT <column names> FROM <table> WHERE <condition>`.

To learn more about sqlite3, visit `tutorialspoint.com/sqlite/`.

3.1 Processing the Data

I recommend getting familiar with tools like python to process your data. A simple set-up would be installing the 'Anaconda' python distribution, opening a jupyter notebook, and reading your data as such:

```
import matplotlib as mpl, numpy as np, sqlite3, matplotlib.pyplot as plt, pandas as pd,
    ↪ seaborn as sns

db = sqlite3.connect("app/data.db")
df = pd.read_sql_query("SELECT␣*␣FROM␣processed", db)
db.close()

sns.set_style("darkgrid", {"legend.frameon": True}) # optional; looks nice
```

To plot for example the correlation between age and concentration performance, you would then simply do:

```
from scipy import stats

def pearson(x, y):
    return stats.pearsonr(x, y)[0]
```

```
fig, ax = plt.subplots()
treatments = ["groupA", "groupB", "groupC"]

for grp in treatments:
    grp_df = df.loc[df['treatment']==grp]
    ax = sns.regplot(x="AGE", y="CP", x_jitter=.2, data=grp_df, label="{} (r = {})
        ↪ ".format(grp, round(pearson(grp_df["AGE"], grp_df["CP"]),3)))

ax.legend()
ax.set(title="Correlation between Age and Concentration Performance", xlabel="Age",
    ↪ ylabel="Concentration Performance")

plt.savefig("raw-age-cp.pdf") # you can also save as png or other formats
plt.show()
```

4 Help!

How do I open a terminal window? On windows, press the windows key and 'R' simultaneously. Then type `cmd`. This is a terminal window. To navigate to the desired folder, type `cd <folder>`, e.g. `cd C:\Documents\d2test`

`npm install` throws an error! This can happen. Read the error message and see if you can figure out what might be wrong. In general, you will probably need to install some type of C compiler. On mac, run `xcode-select --install` from a terminal window. On windows, search 'Visual Studio Build Tools' on the internet.

The website looks weird. You are likely using an outdated web browser such as internet explorer. Consider using Google Chrome (Safari on Mac) or Firefox.

I need something else. Feel free to contact me at `henrik@laxhuber.com`.