My title

Firstname Lastname

optional@email.com
My optional Institute
My optional University

Abstract Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua.

Keywords: Lorem, Ipsum

1 Story Line

- Test Time Generation - Directed Random is faster in regards to AVM and AVM-Defaults.
- Coverage - As Directed Random is faster it has the same coverage as the rest of the techniques.
- Mutation Score - Directed Random has the higher or similar mutation score comparing to AVM and AVM-Defaults.

2 RQs

2.1 Test Time Generation

How efficent Directed Random in regard of test time generation comparing to the other data generators?

2.2 Coverage

How do the number of test requirements generated by the coverage criteria differ depending on the data generation technique being used, and how successfully can test cases be automatically generated to satisfy them?

2.3 Fault-Finding Effectiveness

How does fault finding effectiveness vary depending on data generation technique used?

3 What Plots ?

3.1 Test time generation (Effiency)

3.1.1 Analyse Wilcox Rank and Effect Size for AVM and Directed Random

To statistically analyze the the new technique we conducted tests for significance with the nonparametric Wilcoxon rank-sum test, using the sets of 30 execution times obtained with a specific DBMS and all techniques **Hitchhicker Guide Ref**. A p-value that less than 0.05 is considered significant. To review practical are significance tests we use the nonparametric A12 statistic of Vargha and Delaney **REF** was used to calculate effect sizes. The A12 determine the average probability that one approach beats another, or how superior one technique compared to the other. We followed the guidelines of Vargha and Delaney in that an effect size is considered to be "large" if the value of A12 is $ < 0:29$ or > 0.71, "medium" if A12 is < 0.36 or > 0.64 and "small" if A12 is < 0.44 or > 0.56. However, is the values of A12 close to the 0.5 value are viewed as there no effect.

When comparing AVM and Directed Random techniques in regard of time we used Mann-Whitney U-test and the A-hat effect size calculations. As Shown in in the following two tables that there is statistically significat differance between Driected Random and AVM, $p \leq 0.05$. Therefore, we reject the null hypothesis that there is no difference between AVM and Directed Random. As p-value near zero, that directed random is faster than AVM in a statistically significant test. Moreover, the A-12 shows that Directed Random has a large effect size when it comes to test generation timing. Which menas that Directed Random is the winner in regard of test generation timing.

p.value	dbms
0	Postgres
0	SQLite
0	HyperSQL

Figure 1 and 2.

3.2 Coverage

Figure 3 and 4.

3.3 Mutation Score

Figure 5, 6 and 7.

- Directed Random has a higher PKColumnA operator kills than AVM-Defaults.
- Primary Key Composite key can be problematic for AV<-Defaults.
- Directed Random has either similar or less (because of coverage) mutation kill compared to AVM.
- AVM has lower coverage than AVM-Defaults however it has a higher mutation score because of empty strings and Composite keys.

	Postgres			SQLite			HyperSQL		
Case Study	DR	AVM	AVMD	DR	AVM	AVMD	DR	AVM	AVMD
ArtistSimilarity	1019.9	1406.4	1079.7	288.6	717.5	444.7	492.4	961.7	598.4
ArtistTerm	2603.4	3103.9	2679.6	327.6	913.9	541.5	555.3	1152.7	718.8
BankAccount	1327.0	1619.6	1588.5	321.9	624.8	574.6	533.7	834.9	756.1
BookTown	8780.0	10940.0	11587.8	2002.3	4908.5	5443.3	2345.7	4634.5	5491.0
BrowserCookies	3215.2	17499.5	5849.8	419.3	15703.0	3234.6	657.6	15044.5	3366.1
Cloc	1149.2	1275.7	1192.6	302.7	408.6	433.2	514.4	631.4	599.4
CoffeeOrders	4429.6	4898.4	4743.9	398.2	849.1	823.6	653.6	1105.7	1078.6
CustomerOrder	7944.4	10622.1	8647.8	545.4	3217.7	1789.3	863.6	3364.1	1870.2
DellStore	4186.7	4955.1	4844.9	483.9	1276.7	1137.1	829.6	1632.1	1562.4
Employee	1045.3	1271.7	1340.6	341.1	592.5	700.6	552.8	815.8	898.7
Examination	4052.1	4938.5	4835.7	488.3	1448.5	1267.8	784.3	1737.6	1569.1
Flights	2483.6	24840.6	5771.7	453.6	24019.3	3904.0	690.0	23295.5	3989.1
FrenchTowns	3022.7	4171.4	3860.3	429.4	1628.6	1938.0	683.0	1941.1	1699.5
Inventory	696.7	751.5	801.8	283.7	353.9	436.5	481.6	555.6	601.7
Iso3166	476.8	544.4	500.5	274.9	353.0	402.7	472.4	547.7	546.4
IsoFlav_R2	5125.8	5685.0	5479.2	432.4	986.1	932.7	754.8	1311.2	1265.6
iTrust	46962.1	85790.2	55277.0	4583.8	47114.7	14124.8	4910.9	47908.5	15985.8
JWhoisServer	4031.4	5154.1	4874.0	550.3	1786.9	1550.1	890.2	2085.7	1876.9
MozillaExtensions	6358.3	7616.2	7339.4	545.5	1647.0	1554.6	863.3	2009.3	1916.8
MozillaPermissions	1077.1	1162.0	1190.4	307.9	404.9	492.0	513.4	608.7	663.4
NistDML181	1553.9	1800.9	1706.7	317.2	617.4	540.4	528.3	834.9	706.2
NistDML182	5737.0	7429.5	6811.2	501.3	2097.4	2087.6	761.8	2363.4	1942.0
NistDML183	1321.5	1438.0	1441.7	295.5	359.8	475.2	510.4	577.0	638.1
NistWeather	1934.3	2642.7	2517.5	481.2	1138.2	1216.9	709.9	1421.5	1312.6
NistXTS748	613.9	659.2	706.9	278.0	331.3	501.4	475.4	534.1	613.9
NistXTS749	1535.2	1807.8	1772.5	326.8	566.2	693.5	546.0	775.9	823.5
Person	677.8	1170.4	1728.4	341.8	866.4	1563.3	546.1	1050.8	1599.0
Products	3135.5	6610.6	7279.2	1310.6	5029.1	5852.2	1519.4	5004.2	5796.4
RiskIt	11696.2	14718.1	12527.7	630.5	3475.7	1987.6	996.6	3619.7	2312.9
StackOverflow	4662.2	4828.1	5010.0	480.6	841.9	1124.2	816.4	1169.1	1474.7
StudentResidence	1429.8	1720.3	1539.8	383.6	750.6	628.4	592.8	965.5	782.6
UnixUsage	11108.9	13308.4	11523.3	524.2	2988.5	1674.1	866.6	3479.7	1925.2
Usda	6231.0	6397.2	6474.4	491.0	1007.2	1027.8	864.7	1400.6	1526.3
WordNet	3635.8	3918.2	3985.0	398.8	665.0	836.0	679.2	967.9	1125.1

Table 1. Table

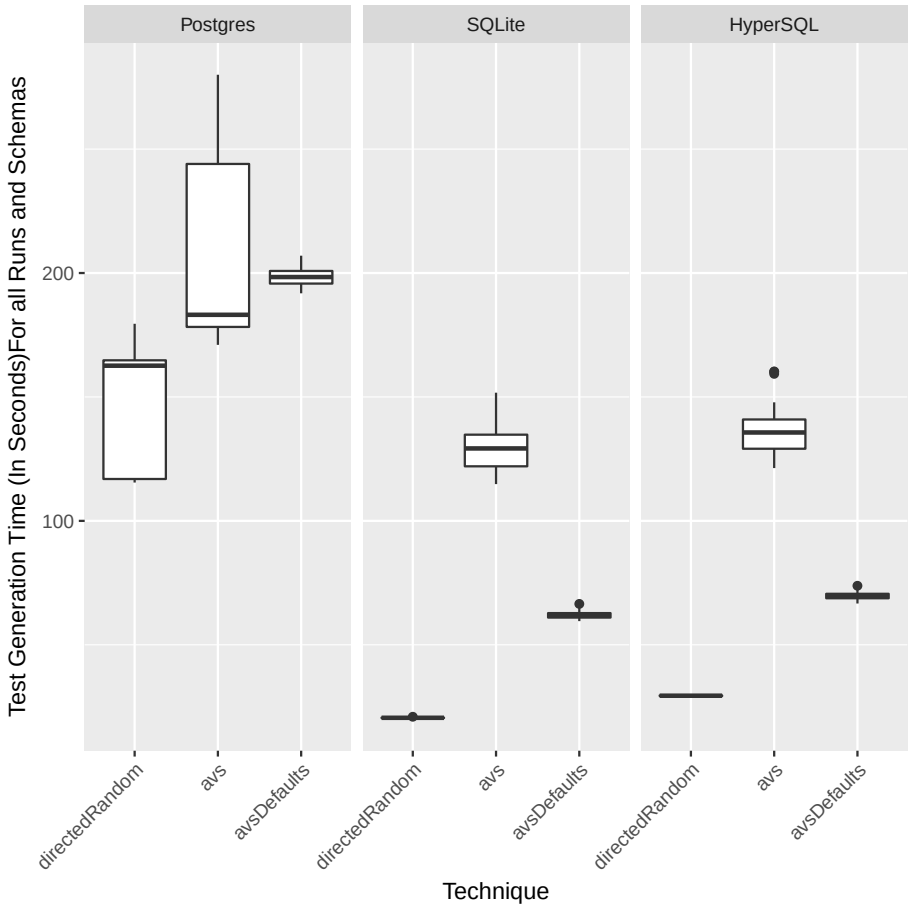

Figure 1. Test generation timing - in seconds. First summing test generation timing for each run then group by data generator and DBMS, then divide test generation timing by 1000 to convert to second.

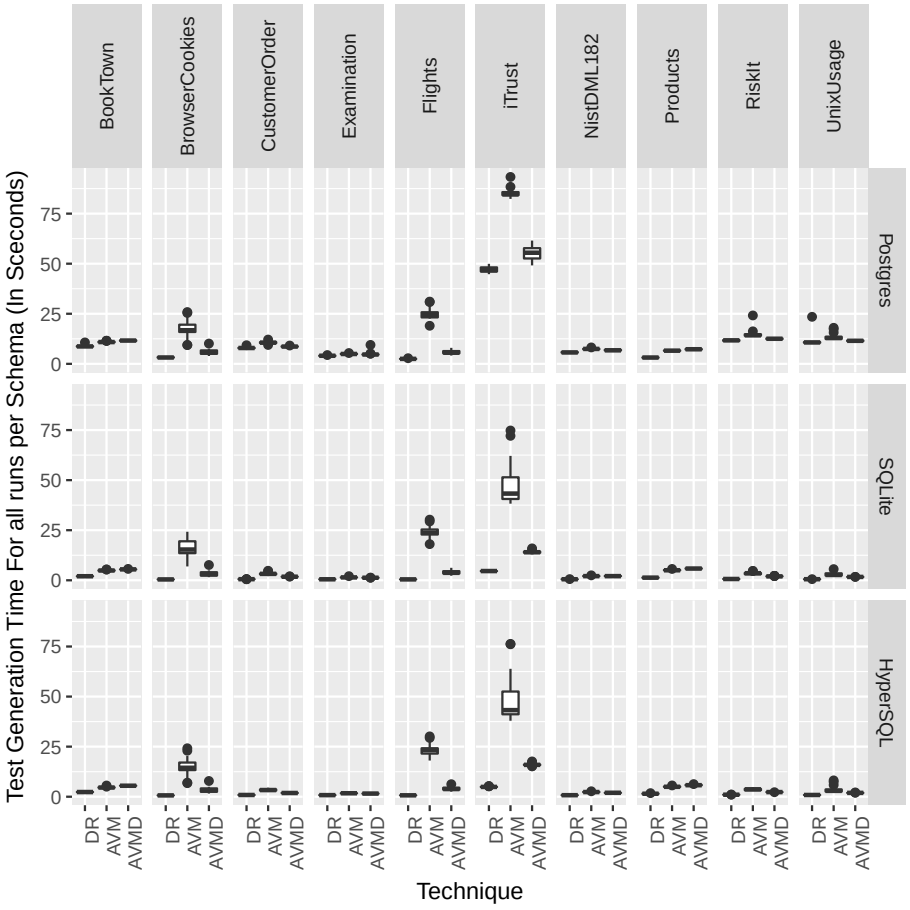

Figure 2. Test generation timing box plot showing some of Game Changer schemas

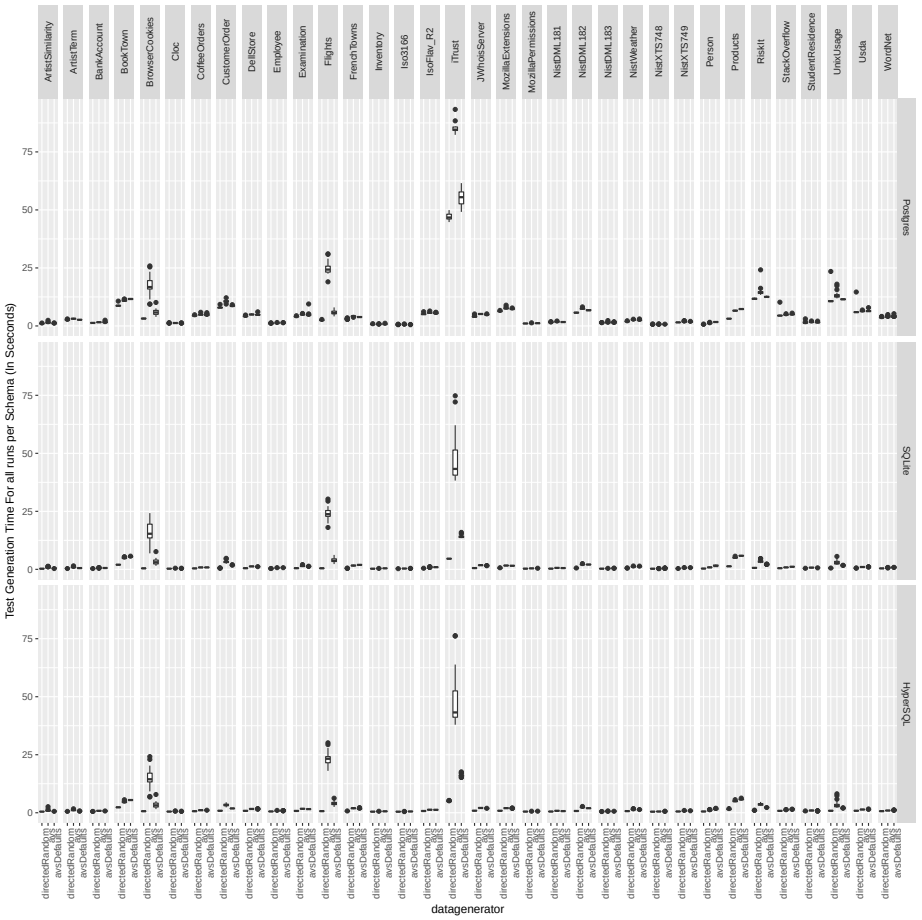

Figure 3. Box plot for Test Generation time for DBMS, techniques and schemas - in precentage. Group by data generator, case study and DBMS, then dividing test generation timning by 1000 to convert to second. No averaging or summing to see the spread of values for all runs per schema

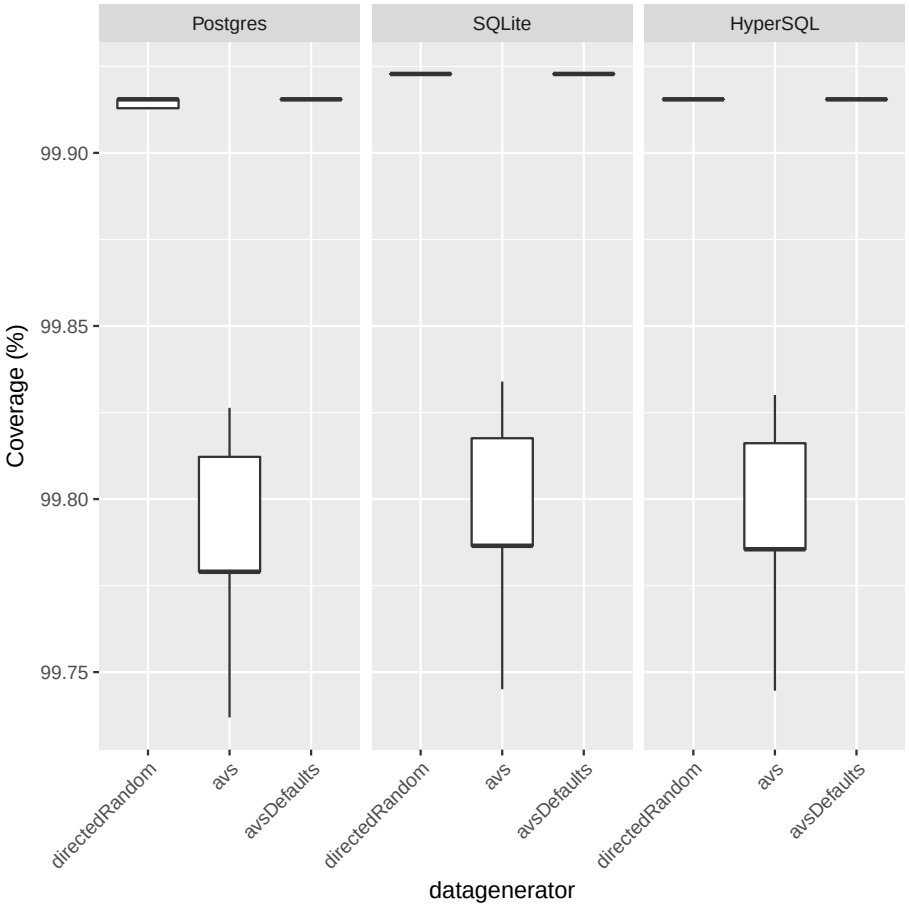

Figure 4. Box plot for Coverage, averaging all schema coverage per run

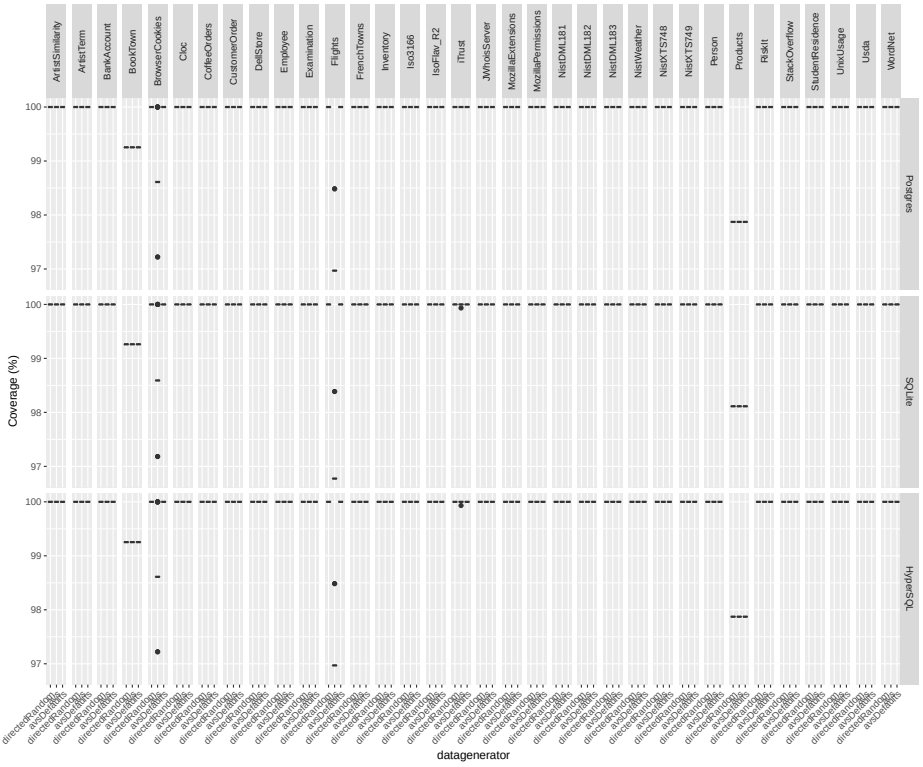

Figure 5. Box plot for Coverage

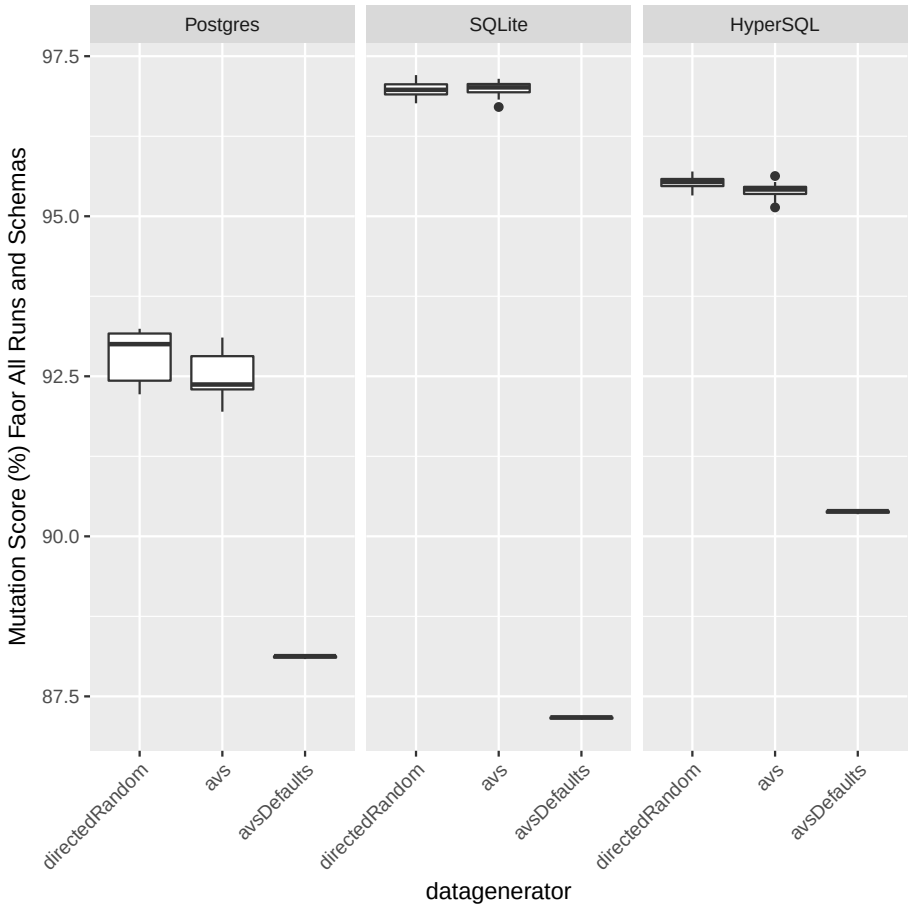

Figure 6. Box plot for mutation scores for DBMSs and techniques - in precentage. Frist I sum all score numerator and denominator per run per DBMS per data generator, then plottig using group by data generator, DBMS and divding the score numerator by denominator multiplying by 100

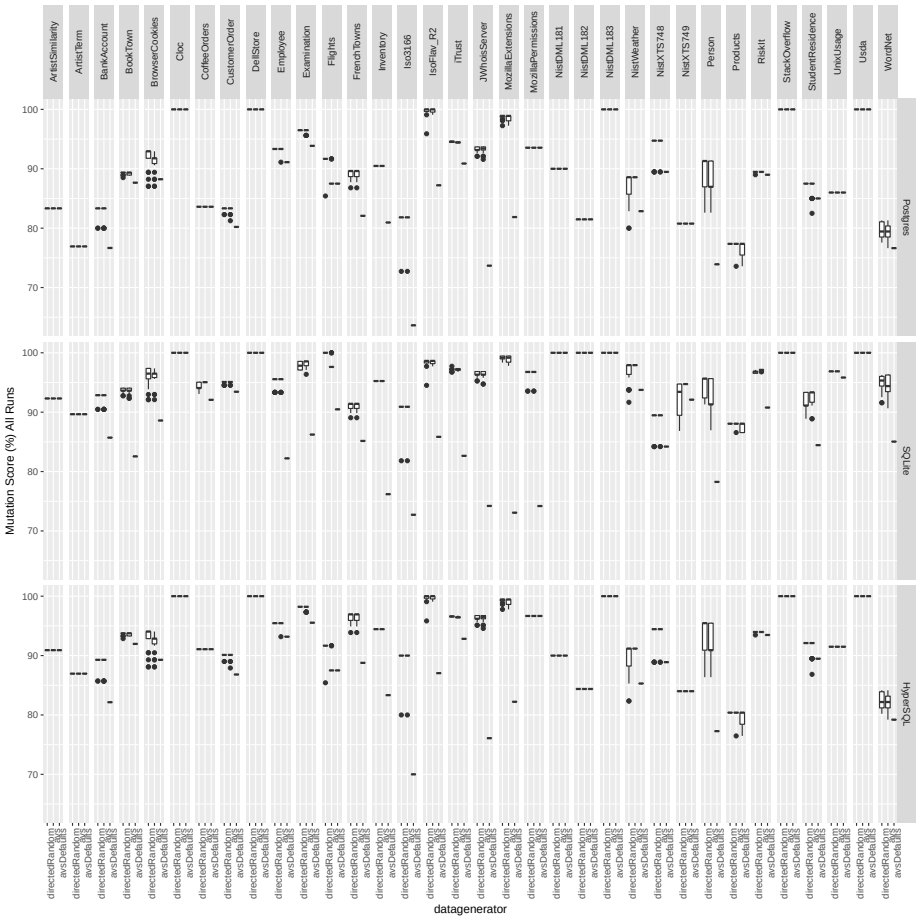

Figure 7. Box plot for mutation scores for DBMS, techniques and schemas - in precentage. Group by data generator, case study and DBMS, then divding the score numerator by denominator multiplying by 100. No averaging or summing to see the spread of values for all runs per schema

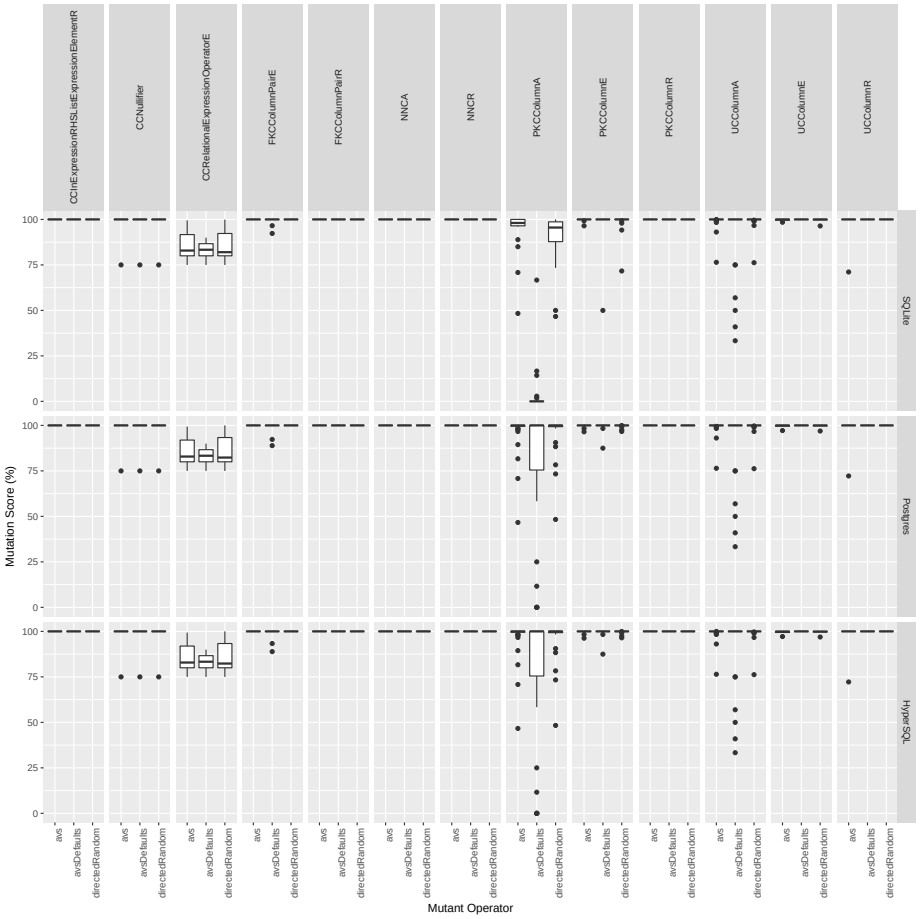

Figure 8. Box Plot of mutant scores in regard of Mutant Operators and DBMSs - in precentage. Using mutanttiming file, I group by generator, DBMS, schema and operator. Then I only select NORMAL mutants, then calculating the percentage of killed mutant by summing all killed divided by (killed mutant plus alive mutant) multiply by 100.