Yahya Alhinai

EE5373

September 17th, 2019

Lab 02

OBJECTIVE FOR THIS LAB:

- We will be conducting various analyze for the data provided. Analyze consist of look for unusual patterns, outliers, and missing data points. Multiple procedures have been conducted for each of data set.

TESTED THE COLUMNS:

Mean:

- Several built-in functions were used to obtain the mean of every column of every processors data type. The following functions were used:
    - mean(): return the mean value for the given data.
    - sapply (): to apply mean for every column of any data table.
    - cbind(): combining all the resulted tables into one big table.
    - na.rm = TRUE: to remove all the NA value from the resulted table

  the following table is the results of obtained mean for every column of all the provided data:

	D1_int92	D1_int95	D1_int00	D1_int06	D1_fp92	D1_fp95	D1_fp00	D1_fp06
nperf	26.5285716	27.0317012	33.0291337	2.497174e+01	26.9106799	18.9024623	3.442799e+01	3.597528e+01
perf	124.2859363	13.3702367	1078.3951111	1.924723e+01	152.7610421	16.8592876	1.217282e+03	1.919988e+01
clock	134.9230769	327.2162162	1750.8554688	2.667382e+03	136.2112676	333.1686047	1.750643e+03	2.684539e+03
threads	1.0000000	1.0270270	1.2304688	1.457286e+00	1.0000000	1.0174419	1.219608e+00	1.466321e+00
cores	1.0000000	1.0000000	1.3398438	2.857843e+00	1.0000000	1.0000000	1.337255e+00	2.797927e+00
TDP	24.9452381	33.8545333	76.3050847	8.808172e+01	25.2365854	33.9471622	7.807895e+01	8.844686e+01
transistors	3.2482692	12.9792241	180.9387402	5.250864e+02	3.3387500	13.6931731	1.783521e+02	5.186663e+02
dieSize	209.5272727	197.7750000	193.3560606	1.985529e+02	211.6078431	204.4074074	1.949618e+02	1.994172e+02
voltage	3.4924074	2.6923246	1.3884600	9.192800e-01	3.4398000	2.6516990	1.400061e+00	9.184072e-01
featureSize	0.5994667	0.3754595	0.1365039	5.721429e-02	0.5957353	0.3750581	1.370000e-01	5.636788e-02
channel	0.5898667	0.3532044	0.0929000	3.502051e-02	0.5851471	0.3523214	9.320188e-02	3.460938e-02
FO4delay	213.9991705	127.9948116	31.8992659	1.213815e+01	212.2049675	127.1621450	3.209939e+01	1.198724e+01
L1icache	13.8730159	46.9880952	66.6509804	3.720408e+01	14.0862069	47.7307692	6.710588e+01	3.689119e+01
L1dcache	18.9473684	69.1219512	87.8117647	3.722449e+01	19.3207547	72.2631579	9.063529e+01	3.689119e+01
L2cache	211.2000000	307.8620690	1647.1231527	3.152245e+03	211.2000000	307.8620690	1.620706e+03	3.225119e+03
L3cache	NaN	NaN	7116.8000000	6.348800e+03	NaN	NaN	7.307636e+03	6.351768e+03

Removing all the NA values is crucial step to implement; Otherwise, if any column has a single NA value, the whole column would be valued to NA when mean() is called.

Variance:

- the same procedures that is implemented to obtain the mean for all of the provided data, variance is conducted similarly. We replace the mean with the following:
    - var(): returns the variance for any set of data provided.

  The following table is the result of the produced results:

	D2_int92	D2_int95	D2_int00	D2_int06	D2_fp92	D2_fp95	D2_fp00	D2_fp06
nperf	5.595413e+02	5.647742e+02	5.058359e+02	2.420109e+02	4.674391e+02	3.337446e+02	5.758009e+02	5.352835e+02
perf	6.099204e+03	1.171280e+02	4.470319e+05	7.070111e+01	1.170468e+04	2.308040e+02	6.200206e+05	6.968141e+01
clock	5.406955e+03	4.570399e+04	8.457664e+05	2.885670e+05	5.455798e+03	4.826071e+04	8.422636e+05	3.191943e+05
threads	0.000000e+00	2.643948e-02	1.780484e-01	2.494290e-01	0.000000e+00	1.723786e-02	1.720550e-01	2.501619e-01
cores	0.000000e+00	0.000000e+00	3.899357e-01	1.432894e+00	0.000000e+00	0.000000e+00	3.897483e-01	1.255829e+00
TDP	2.262284e+02	4.702374e+02	1.347287e+03	1.031627e+03	2.282299e+02	4.760267e+02	1.272512e+03	1.097140e+03
transistors	8.615062e+00	6.877414e+02	3.318425e+04	8.045071e+04	9.179207e+00	7.564154e+02	3.349787e+04	6.684316e+04
dieSize	4.906291e+03	1.628061e+04	8.123391e+03	7.684249e+03	4.827763e+03	1.672163e+04	8.185483e+03	8.001590e+03
voltage	4.592979e-01	4.352027e-01	1.572234e-01	3.765704e-02	3.996877e-01	3.844542e-01	1.537692e-01	3.294905e-02
featureSize	2.706728e-02	2.840536e-02	3.816651e-03	3.108051e-04	2.543676e-02	2.873391e-02	3.693031e-03	2.708692e-04
channel	3.048512e-02	3.180856e-02	5.163258e-03	7.263875e-05	2.912087e-02	3.200476e-02	4.851049e-03	6.160054e-05
FO4delay	4.051730e+03	4.057029e+03	5.729503e+02	1.103454e+01	3.848343e+03	4.110854e+03	5.587693e+02	9.499346e+00
L1icache	4.703385e+02	8.845772e+03	1.227609e+04	2.236094e+02	5.077995e+02	9.452366e+03	1.301066e+04	2.126183e+02
L1dcache	1.516872e+03	3.596791e+04	4.932784e+04	2.114981e+02	1.630607e+03	3.867755e+04	5.274969e+04	2.012850e+02
L2cache	2.940018e+04	3.524096e+04	3.679166e+06	1.106768e+07	2.940018e+04	3.524096e+04	3.692850e+06	1.117310e+07
L3cache	NA	NA	2.181329e+07	8.710779e+06	NA	NA	2.003257e+07	8.838253e+06

  All NA values were removed for the same reason explained in the last section. One thing worth mentioning is that L2chahe did not exists in processors in the year of 1992 and 1995; therefore, its labeled as NA because no single value is found in the those column.

Minimum:

- similar to Mean and variance sections, set of built-in function were used to obtain the desirable value. To get the minimum value the following function was used:
    - min(): out of all the data points given to this function, it retunes the minimum value of the data.

  The following table shows the value returned after calling min function after NA value were not computed:

	D3_int92	D3_int95	D3_int00	D3_int06	D3_fp92	D3_fp95	D3_fp00	D3_fp06
nperf	0.000	0.00	0.00000	0.000	0.000	0.00	0.00000	0.000
perf	36.700	1.06	96.50793	5.750	18.100	1.14	87.54153	6.220
clock	48.000	40.00	250.00000	1200.000	50.000	40.00	200.00000	1200.000
threads	1.000	1.00	1.00000	1.000	1.000	1.00	1.00000	1.000
cores	1.000	1.00	1.00000	1.000	1.000	1.00	1.00000	1.000
TDP	3.000	3.90	11.00000	20.000	3.000	3.90	13.00000	20.000
transistors	0.580	0.58	5.10000	0.000	0.580	0.58	5.10000	105.000
dieSize	76.000	17.00	81.00000	77.000	76.000	17.00	81.00000	77.000
voltage	2.285	1.65	0.80000	0.600	2.285	1.65	0.85000	0.650
featureSize	0.290	0.18	0.04500	0.032	0.290	0.18	0.06500	0.032
channel	0.250	0.10	0.03500	0.018	0.250	0.10	0.03500	0.018
FO4delay	90.000	36.00	10.40909	6.480	90.000	36.00	10.40909	6.480
L1icache	1.000	1.00	8.00000	12.000	1.000	1.00	8.00000	12.000
L1dcache	0.000	0.00	8.00000	16.000	0.000	0.00	8.00000	16.000
L2cache	96.000	96.00	96.00000	256.000	96.000	96.00	96.00000	256.000
L3cache	Inf	Inf	1536.00000	2048.000	Inf	Inf	1536.00000	2048.000

Maximum:

- Like minimum section, maximum, built-in function, is used here to return back the maximum value in every column. To get the maximum value the following function was used:
    - Max(): return back the highest value from the data given.

After NA were removed from data computed, the following table is describing the maximum value of all columns of all set of data:

	D4_int92	D4_int95	D4_int00	D4_int06	D4_fp92	D4_fp95	D4_fp00	D4_fp06
nperf	5.595413e+02	5.647742e+02	5.058359e+02	2.420109e+02	100.0	100.0	100.00	100.000
perf	6.099204e+03	1.171280e+02	4.470319e+05	7.070111e+01	518.5	84.3	3369.00	42.300
clock	5.406955e+03	4.570399e+04	8.457664e+05	2.885670e+05	350.0	1000.0	3800.00	5000.000
threads	0.000000e+00	2.643948e-02	1.780484e-01	2.494290e-01	1.0	2.0	2.00	2.000
cores	0.000000e+00	0.000000e+00	3.899357e-01	1.432894e+00	1.0	1.0	4.00	6.000
TDP	2.262284e+02	4.702374e+02	1.347287e+03	1.031627e+03	56.0	100.0	165.00	160.000
transistors	8.615062e+00	6.877414e+02	3.318425e+04	8.045071e+04	15.0	210.0	1328.00	1328.000
dieSize	4.906291e+03	1.628061e+04	8.123391e+03	7.684249e+03	335.0	1215.0	467.00	435.000
voltage	4.592979e-01	4.352027e-01	1.572234e-01	3.765704e-02	5.0	5.0	2.50	1.350
featureSize	2.706728e-02	2.840536e-02	3.816651e-03	3.108051e-04	1.0	1.0	0.35	0.130
channel	3.048512e-02	3.180856e-02	5.163258e-03	7.263875e-05	1.0	1.0	0.35	0.053
FO4delay	4.051730e+03	4.057029e+03	5.729503e+02	1.103454e+01	360.0	360.0	126.00	25.200
L1icache	4.703385e+02	8.845772e+03	1.227609e+04	2.236094e+02	128.0	512.0	768.00	64.000
L1dcache	1.516872e+03	3.596791e+04	4.932784e+04	2.114981e+02	256.0	1024.0	1536.00	64.000
L2cache	2.940018e+04	3.524096e+04	3.679166e+06	1.106768e+07	512.0	512.0	16384.00	16384.000
L3cache	*NA*	*NA*	2.181329e+07	8.710779e+06	-Inf	-Inf	16384.00	16384.000

Sorting:

- The RStudio interface has a build-in feature in the table view where you can sort each column at time. By having elements sorted, it is easy to look for outliers if there is any. An example of part of a sorted column is shown below. The table is sorting “nperf” in the set of data “int06.dat”. we can see the non-descending order for npref column with all the rest information next to it.

4	0.000000	5.750000	1200	1	1	NA	NA	NA	NA	0.090	0.053	19.08000	64	64	512	NA
37	2.183164	6.930000	1800	1	2	NA	NA	NA	NA	0.090	0.053	19.08000	64	64	512	NA
9	2.849214	7.290000	1600	1	2	20.0	NA	NA	1.0750	0.065	0.035	12.60000	64	64	512	NA
38	3.570720	7.679974	2000	1	2	31.0	153.8	126	1.1250	0.065	0.035	12.60000	64	64	512	NA
150	4.125414	7.979786	2600	2	2	95.0	1328.0	435	1.1000	0.065	0.035	10.84737	16	16	4096	4096
131	4.310823	8.080000	2660	2	2	95.0	230.0	206	1.2000	0.090	0.053	19.08000	12	16	2048	NA
151	6.028387	9.008343	3000	2	2	95.0	1328.0	435	1.1000	0.065	0.035	10.84737	16	16	4096	4096
12	6.049954	9.020000	2000	1	1	89.0	106.0	193	1.4000	0.130	0.070	25.20000	64	64	1024	NA
139	6.160799	9.079912	1867	1	1	NA	NA	NA	NA	0.090	0.053	19.08000	32	32	2048	NA
133	6.475486	9.250000	3000	2	2	95.0	376.0	162	1.2000	0.065	0.035	10.89022	12	16	4096	NA
132	6.586494	9.310000	2800	2	2	95.0	376.0	162	1.2000	0.065	0.035	10.89022	12	16	4096	NA
97	7.510800	9.809587	1830	2	2	31.0	151.0	90	1.1625	0.065	0.035	11.02593	32	32	2048	NA
134	7.585569	9.850000	3000	2	2	95.0	376.0	162	1.2000	0.065	0.035	10.89022	12	16	4096	NA
157	7.880612	10.009471	1600	1	4	80.0	582.0	286	1.0000	0.065	0.035	10.40909	32	32	8192	NA
152	7.904704	10.022493	3200	2	2	150.0	1328.0	435	1.1000	0.065	0.035	10.84737	16	16	8192	8192
5	8.048104	10.100000	2600	1	2	110.0	233.0	199	1.3000	0.090	0.053	19.08000	64	64	1024	NA
98	8.048104	10.100000	2000	1	2	NA	NA	NA	NA	0.065	0.035	11.43574	32	32	2048	NA
144	8.227722	10.197084	1600	2	2	65.0	291.0	143	1.0000	0.065	0.035	10.40909	32	32	4096	NA
41	8.291602	10.231611	2100	2	2	NA	276.0	389	1.2000	0.130	NA	25.20000	64	32	1920	NA
135	8.603145	10.400000	3200	2	2	130.0	376.0	162	1.2000	0.065	0.035	10.89022	12	16	4096	NA
121	9.101998	10.669630	1400	1	2	NA	NA	NA	NA	0.090	0.053	19.08000	16	16	256	6144
140	9.158187	10.700000	2266	1	1	NA	NA	NA	NA	0.090	0.053	19.08000	32	32	2048	NA
36	9.528215	10.900000	2000	1	4	65.0	463.0	283	1.0750	0.065	0.035	12.60000	64	64	512	2048
47	9.528215	10.900000	1866	1	1	NA	NA	NA	NA	0.065	0.035	11.43574	32	32	512	NA
136	9.528215	10.900000	3400	2	2	130.0	376.0	162	1.2000	0.065	0.035	10.89022	12	16	4096	NA
166	9.713228	11.000000	1860	1	4	50.0	582.0	286	1.1000	0.065	0.035	11.22549	32	32	8192	NA
122	9.797153	11.045361	1600	1	2	NA	NA	NA	NA	0.090	0.053	19.08000	16	16	256	6144
153	9.801125	11.047508	3400	2	2	150.0	1328.0	435	1.1000	0.065	0.035	10.84737	16	16	16384	16384
14	9.898242	11.100000	2800	1	1	92.6	106.0	115	1.3500	0.090	0.053	19.08000	64	64	1024	NA
46	9.898242	11.100000	2000	1	1	NA	NA	NA	NA	0.065	0.035	11.43574	32	32	512	NA
50	9.935172	11.119960	1800	1	2	65.0	167.0	111	0.8500	0.065	0.035	10.40909	32	32	2048	NA
145	10.268270	11.300000	1860	2	2	65.0	291.0	143	1.0000	0.065	0.035	10.40909	32	32	4096	NA
158	10.323750	11.329987	1860	1	4	80.0	582.0	286	1.0000	0.065	0.035	10.40909	32	32	8192	NA
6	10.453284	11.400000	2800	1	2	125.0	243.0	220	1.3500	0.090	0.053	19.08000	64	64	1024	NA
137	10.453284	11.400000	3600	2	2	130.0	376.0	162	1.2000	0.065	0.035	10.89022	12	16	4096	NA
13	10.638298	11.500000	3000	1	1	NA	NA	NA	NA	0.090	0.053	19.08000	64	64	1024	NA

Fraction of NA values:

- This section is meant to identify the percentage values of the data missing (NA) to identify if there is enough data to draw a conclusion. The following function were used to compute the missing value:
    - colMeans(is.na()): this is actually two different functions. Is.na() will return the percentage of data points that is value as NA out of all data points. colMeans() will apply whatever function (in this case is.na() function) is pass into it to be applied to all column.

The following table provide the parentage of the data missing in every column of all the data sets:

	D4_int92	D4_int95	D4_int00	D4_int06	D4_fp92	D4_fp95	D4_fp00	D4_fp06
nperf	5.595413e+02	5.647742e+02	5.058359e+02	2.420109e+02	100.0	100.0	100.00	100.000
perf	6.099204e+03	1.171280e+02	4.470319e+05	7.070111e+01	518.5	84.3	3369.00	42.300
clock	5.406955e+03	4.570399e+04	8.457664e+05	2.885670e+05	350.0	1000.0	3800.00	5000.000
threads	0.000000e+00	2.643948e-02	1.780484e-01	2.494290e-01	1.0	2.0	2.00	2.000
cores	0.000000e+00	0.000000e+00	3.899357e-01	1.432894e+00	1.0	1.0	4.00	6.000
TDP	2.262284e+02	4.702374e+02	1.347287e+03	1.031627e+03	56.0	100.0	165.00	160.000
transistors	8.615062e+00	6.877414e+02	3.318425e+04	8.045071e+04	15.0	210.0	1328.00	1328.000
dieSize	4.906291e+03	1.628061e+04	8.123391e+03	7.684249e+03	335.0	1215.0	467.00	435.000
voltage	4.592979e-01	4.352027e-01	1.572234e-01	3.765704e-02	5.0	5.0	2.50	1.350
featureSize	2.706728e-02	2.840536e-02	3.816651e-03	3.108051e-04	1.0	1.0	0.35	0.130
channel	3.048512e-02	3.180856e-02	5.163258e-03	7.263875e-05	1.0	1.0	0.35	0.053
FO4delay	4.051730e+03	4.057029e+03	5.729503e+02	1.103454e+01	360.0	360.0	126.00	25.200
L1icache	4.703385e+02	8.845772e+03	1.227609e+04	2.236094e+02	128.0	512.0	768.00	64.000
L1dcache	1.516872e+03	3.596791e+04	4.932784e+04	2.114981e+02	256.0	1024.0	1536.00	64.000
L2cache	2.940018e+04	3.524096e+04	3.679166e+06	1.106768e+07	512.0	512.0	16384.00	16384.000
L3cache	*NA*	*NA*	2.181329e+07	8.710779e+06	-Inf	-Inf	16384.00	16384.000

Table() distributions:

- table() is a built-in function that shows how frequent a specific value shows in every data set. This is useful in identifying if there is a tend or a value that is mostly repeated. An example is the following table which describe the frequency of values in every column of data set “int06.dat”:

table_int06	list [16]	List of length 16
nperf	integer [182] (S3: table)	1 1 1 1 1 1 ...
perf	integer [182] (S3: table)	1 1 1 1 1 1 ...
clock	integer [52] (S3: table)	1 1 6 1 4 1 ...
threads	integer [2] (S3: table)	108 91
cores	integer [6] (S3: table)	10 107 3 78 5 1
TDP	integer [31] (S3: table)	1 1 1 4 2 9 ...
transistors	integer [35] (S3: table)	1 1 3 3 1 4 ...
dieSize	integer [33] (S3: table)	1 13 6 3 1 14 ...
voltage	integer [26] (S3: table)	4 20 2 3 1 1 ...
featureSize	integer [5] (S3: table)	17 81 75 21 2
channel	integer [6] (S3: table)	17 11 2 143 21 1
FO4delay	integer [18] (S3: table)	17 11 27 2 2 2 ...
L1icache	integer [4] (S3: table)	9 9 136 42
L1dcache	integer [3] (S3: table)	18 137 41
L2cache	integer [13] (S3: table)	48 25 14 1 19 7 ...
L3cache	integer [8] (S3: table)	8 2 18 11 24 2 ...

IDENTIFIED AND DESCRIBED UNUSUAL COLUMNS:

Mean:

- The clock seems to grow exponentially between the years 1992 to 2006 for integer and floating point. This is correlated with the number of transistors where it grows exponentially.
- The voltage dropped significantly for all possessors from 3.5V in 1992 to 0.9V in 2006.
- In 1992 and 1995, L3 did not exist. It started in the year 2000.
- dieSize stayed the same in between the year 1992 to 2006 for all possessors.

Variance:

- The most widely distributed result is in transistors section. In 1992, the variance is within 1 digit number. In 1995, the variance is 3-digit number. In 2000 and 2006, the variance within 5-digit number.
- The variance in channel decreases drastically. In 1992, it was 0.03; whereas in 2006, the variance was 0.00007 for integer possessors. Meaning the difference in channel between two possessors is diminishing as time goes by. The same trend found in floating point possessors.

Minimum:

- The most appealing observation goes to the clock. The minimum speed of the clock was 48 in 1996. On the other hand, minimum speed of the clock was 1200. The minimum speed of the clock increased 25 times between the year 1992 and 2006

Maximum:

- The most interesting thing to notice is the cashed L1, and L3. As L3 cashed emerged in 2000, there was a drastic deceased in the maximum size of the L1 cashes. The decrease is in the maximum size of L1 caches is 55 times between the year 2000 and 2006 for integer and 12 time decreases for the floating point possessors.
-

Sort the column to look for outliers or unusual patterns:

- From the sorted columns, the most noticeable column is clock. It seems the most preferred clock speed is 3000 across all possessors and throughout the years 1992 to 2006.

Compute the fraction of NA values to see if there are enough values:

- Cashes have the highest NA fraction especially in L2 and L3 in the year of 1992 and 1995 because the face that they were new technology and they were not widely implemented.
- Also TDP has very high NA fraction.
- Otherwise, it seems that there are enough values to look for patterns.

Use the table() function to determine if the distribution of values appears to have any anomalies:

- The most important observation to be made is in the core column. As we know, multi-core processors have started in the early 2000s. We can see that all processors before 2000 were single core processors. In 2000, processors started to have dual and quadrat cores but single-core processors were still the most widely used. In 2006, single- core processors were the least to be used as dual-core processors were the predominant as well as 4-core processors.
- The voltage has an interesting trend. The most common used voltage in the year of 1992 and 1995 was 3.3V. Then it dropped to 1.2V the most common voltage in the year 2000. In 2006, the most widely used voltage was 0.85V.

DESCRIBED ANOMALIES IN THE DATA:

- The missing values (NA) from the origin data frame is an anomaly. We are not able to compute variance, mean, minimum, maximum, sorting, and distributions of the value with the present of NA values. This is was a major issue because if a single element in a column was missing then the whole column will be disregarded.

- Another thing showed up when analyzing the data was the warnings when trying to call min/max function to compute a specific column. It shows inf/-inf warning.

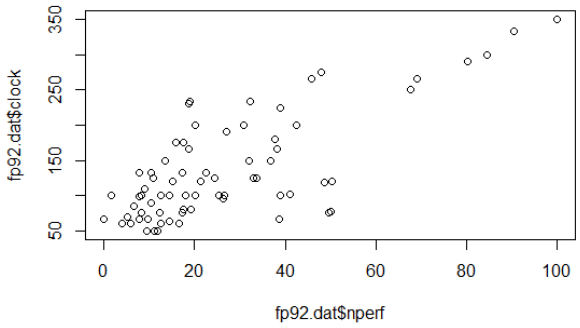
EXPLAINED HOW TO FIX ANOMALOUS DATA:

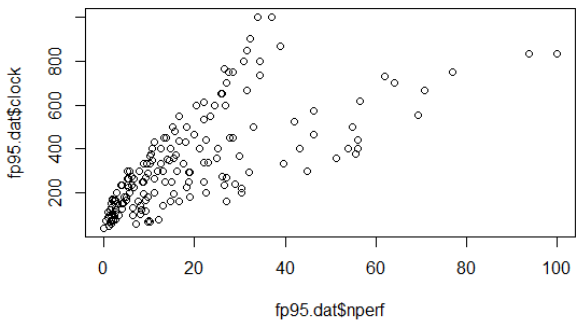
- To get by this issue of anomalous data, we added “na.rm = TRUE” for every function call. This will ignore the missing value as they are not part of the table. It will only take into account the non-missing value.

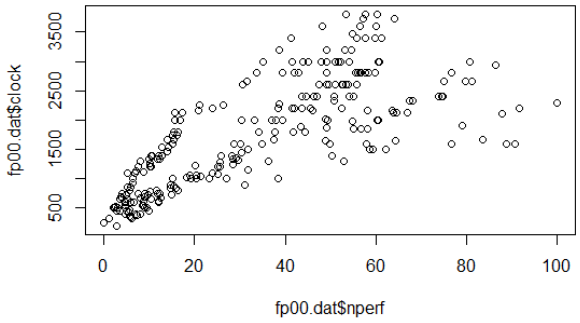
- To fix the warring when min and max function is called the instruction na.rm = TRUE” was assigned to all of function call which will be produce in the resulting data frame when I executed the function. This step will replace the non-existing value with “1” to all unvalued elements.

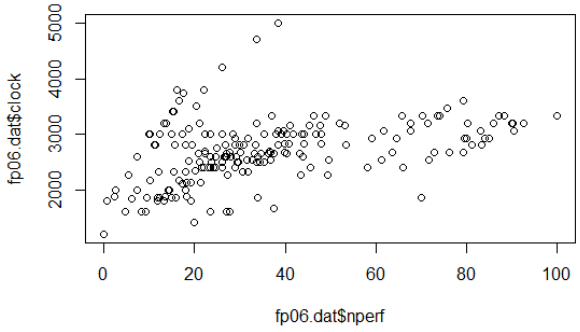
YOU MUST INCLUDE ONE INTERESTING TYPE OF ANALYSIS BEYOND THE LIST ABOVE:

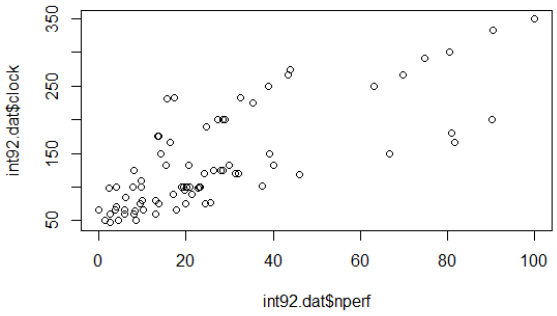
- An interesting type of analysis I notices was the performance vs the clock speed. Before the real use of multi-core processors in the years 1992 to 2000, the clock speed was linearly linked to the performance. In the year of 2006, the increase in clock speed did not respond to a linear increase in the performance. The saturation of performance is because of the shit to multi-core processors.
- The Graph for performance vs clock for all given processors from the year 1992 to 2006 are shown below.

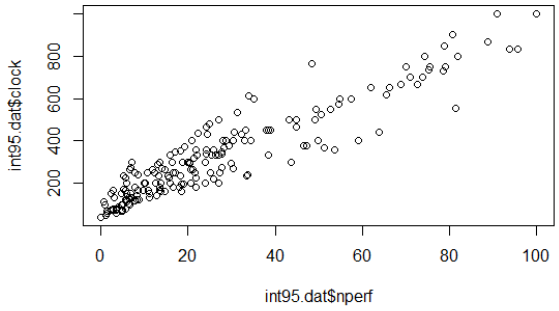


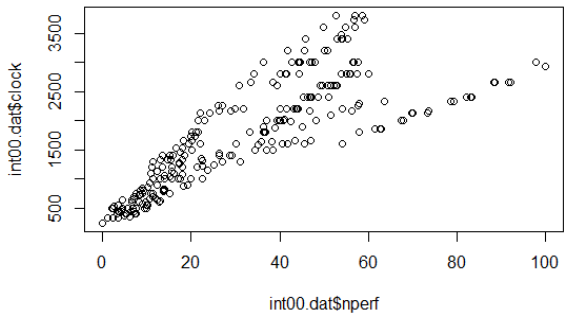


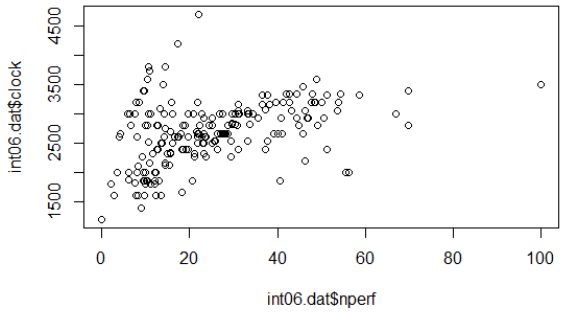

fp00.dat$clock
500
1500
2500
3500
0
20
40
60
80
100
fp00.dat$nperf


fp06.dat$clock
2000
3000
4000
5000
0
20
40
60
80
100
fp06.dat$nperf


int92.dat$clock
50
150
250
350
0
20
40
60
80
100
int92.dat$nperf


int95.dat$clock
200
400
600
800
0
20
40
60
80
100
int95.dat$nperf


int00.dat$clock
500
1500
2500
3500
0
20
40
60
80
100
int00.dat$nperf


int06.dat$clock
1500
2500
3500
4500
0
20
40
60
80
100
int06.dat$nperf