The Manual of VEP annotation pipeline

Icahn School of Medicine at Mount Sinai
Minju Kim

<Workflow overview of the pipeline>

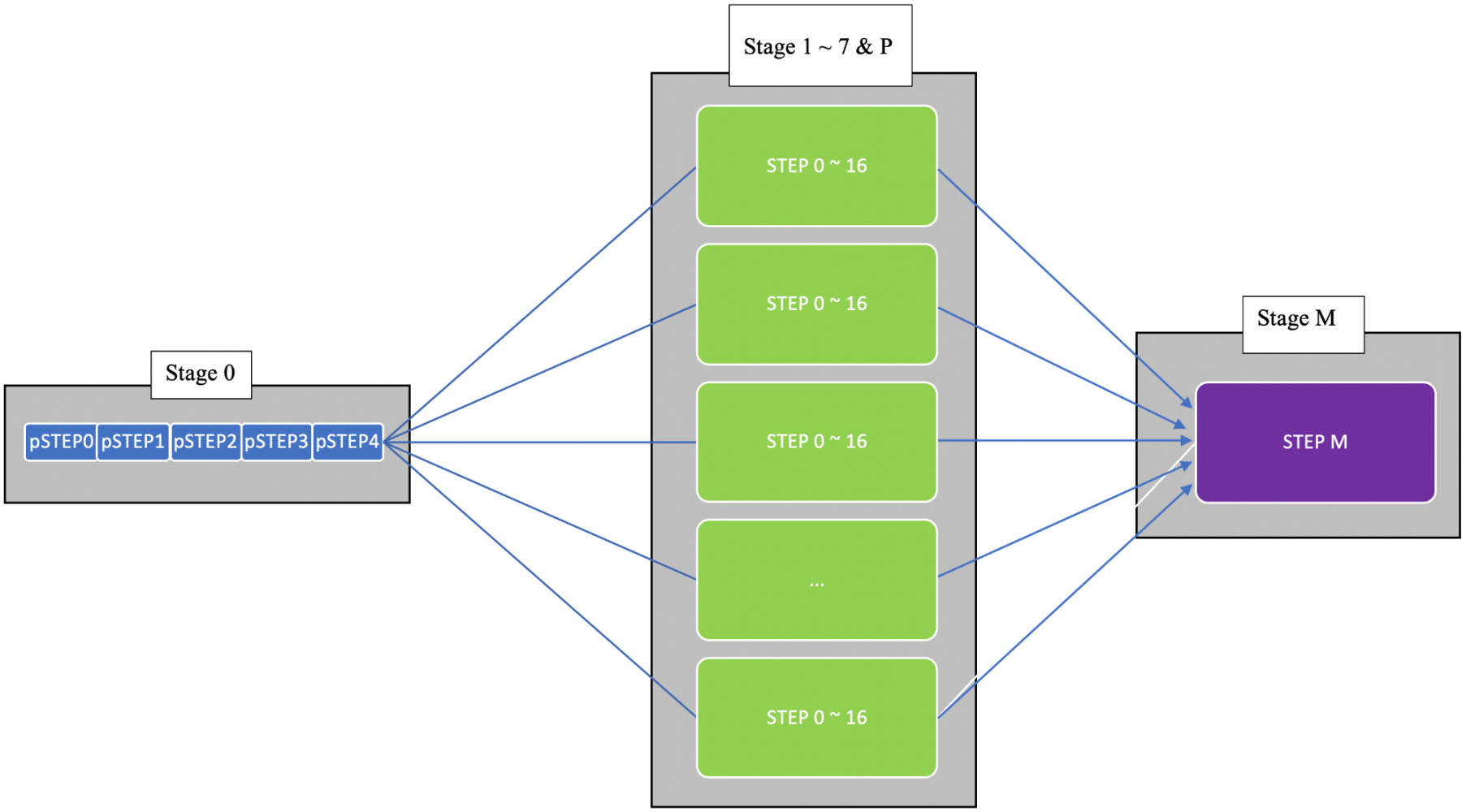


Fig 1. The workflow of VEP annotation pipeline (from Stage 0 to Stage M)

The flowchart above represents the workflow applies to the vcf containing more than 100,000 variants. Stage 0 performs the roles of vcf split, its processing and preparing initiator which initiates the downstream stages: 1 ~ 7 and Post(P). After completing these stages, STEP M should be taken to concatenate all the results obtained from Stage P.

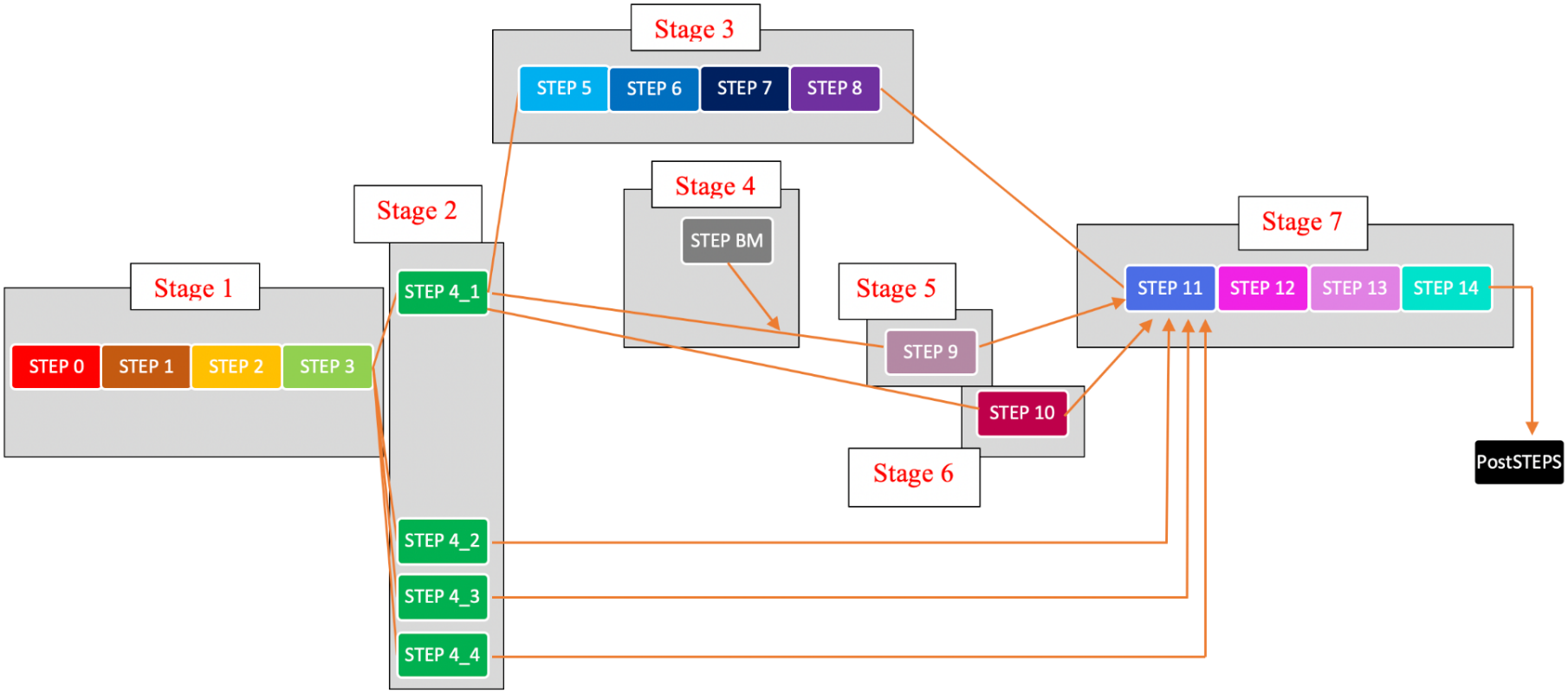


Fig 2. The workflow of VEP annotation pipeline (from Stage 1 to Stage 7)

Each STEP belonging to the stages must be executed sequentially except the total of 4 STEPs in Stage 2. Those STEPs can be executed simultaneously because all of them require the output from STEP3 as the inputs. Also, Stage 4 which contains STEP BM is not necessarily included in the pipeline. The output from that stage has already been prepared in sources. Thus, a user can use the ready-made output as the input of the next stage: Stage 5.

The Manual of VEP annotation pipeline

1.The link to the scripts: https://gitlab.com/itan-lab/gof-lof-classifier/-/tree/cigdem_feat/annotations/Minju/VEP(NEW)
2. The sample results on HPC(Minerva): /sc/arion/projects/Itan_lab/minju/annotations/test
3. The sample results of the pipeline including pre-steps on HPC(Minerva): /sc/arion/projects/Itan_lab/minju/annotations/test/split
4. Path to the scripts on HPC(Minerva): /sc/arion/projects/Itan_lab/minju/annotations/sources
5. Path to the input files on HPC(Minerva): /sc/arion/projects/Itan_lab/minju/annotations/sources/vep_input
6. How to use the automated pipeline

0) Diagnostic Stage (Counting the number of variants of the vcf file which will be used as the input)

<1> Count the number of variants contained in the vcf file using bcftool. The script, 'var_count.sh' is in the directory: /sc/arion/projects/Itan_lab/minju/annotations/sources

1) The Pipeline including the pre-steps (recommended to use when the number of variants of the vcf is greater than 100,000)

<1> Copy all the Scripts: the 5 of "pSTEP scripts (pSTEP0 ~ pSTEP4)", "VEP_ALL_IN_ONE.lsf", the 17 of "lsf files(STEP1 ~ STEP16 and STEPM)", and the 15 of python scripts (STEP3 ~ STEP16) into the user's working directory.

<2> Open the "pSTEP0.sh" file and input the three initial information: 1. Working directory(no "/" at the end of the path); 2. Input vcf file name (only the file name without the directory); 3. The label of the work (e.g. clinvar, gnomAD, and so forth)

<3> Run the script, "VEP_ALL_IN_ONE.lsf" (bsub < VEP_ALL_IN_ONE.lsf)

```
bsub < VEP_ALL_IN_ONE.lsf
```

2) The Pipeline from Stage 1 (recommended to use when the number of variants of the vcf is less than or equal to 100,000)

<1> Copy all the Scripts (1 sh, 17 lsf, and 15 py) into the user's working directory.

<2> Open "STEP0.sh" file and input the three initial information: 1. Working directory; 2. Input vcf file name (only the file name without the directory); 3. The label of the work (e.g. clinvar, gnomAD, and so forth)

<3> Run the script, "vep_all_the_way.lsf"

```
bsub < vep_all_the_way.lsf
```

<4> Run the scripts of Post stages (STEP15, STEP16, and so on) after checking the result file, "vep_{project_label}_14.csv". These steps can be customized according to the user's purpose(s). When selecting the annotation columns, please refer to the sheet, "VEP annotation columns info" with the online tool links here: https://docs.google.com/spreadsheets/d/1m_d_3SAmBzjNek1-SkTt-iuAqBfWaknH/edit?usp=sharing&ouid=102546441492054213912&rtpof=true&sd=true

7. The directory of standard output and error: /path/to/the/working/directory/std and /path/to/the/working/directory/split folder/std
Every standard output and error files per step will be saved in this folder so that the user can easily keep track of the step number the pipeline is on. The directory, "std" will be automatically created while executing the pipeline.

* {label} – the user can use any unique label according to his/her project

Table 1. VEP annotation pipeline stepwise work details

Stage	STEP	Step (old)	Input	Output	Description	Time (per variant) (sec)	Considerations
<Stage 0 (preSTEPs)> 1. If your input vcf file contains more than 100,000 variants, it is highly recommended to start from this stage to reduce the amount of time for annotating. Otherwise, it is recommended to directly start from Stage 1. 2. The entire Stages from 0 to P(postSTEPs) & M(Merging the results) has been automated. Thus, the user can get the result(s) by executing a single control lsf script (VEP_ALL_IN_ONE.lsf)							
	0	NA	Initial Input 1. Working directory 2. the name of vcf file 3. unique label of the project	NA			
	1	NA	INPUT VCF pSTEP0.sh	splitted vcf files: {label}_###.vcf in the each of the directory: {label}_###	Input essential information	0.00003	Use the maximum nodes and memory
	2	NA	pSTEP0.sh and the results of pSTEP1.lsf	each of STEP0.sh in each of the folder: {label}_###	this is required for running Stage 1 ~ 7 & P & M		
	3	NA	pSTEP0.sh vep_all_the_way.lsf (from the sources)	vep_{label}_###.lsf in each of the folder: {label}_###			
	4	NA	pSTEP0.sh vep_{label}_###.lsf (the initiators)	execution of Stage 1 ~ 7 & P in each folder: {label}__###	each initiator will be executed in each directory independently		

<Stage 1 ~ 7 & P> If your input vcf file contains less than or equal to 100,000 variants, it is required to take steps from here							
1	0	NA	Initial Input 1. Working directory 2. the name of vcf file 3. unique label of the project	NA	Input essential information		1. All the shell, lsf and python scripts should be copied into the working directory 2. Create a directory, 'std'
	1	1_1	INPUT VCF	vep_{label}_1.vcf vep_{label}_1.vcf.gz vep_{label}_1.vcf.gz.tbi vep_{label}_1.vcf_warnings.txt	HGMD 38 Annotation	0.00944325	Multiple nodes
	2	1_2	vep_{label}_1.vcf	vep_{label}_2.tsv vep_{label}_2_header.txt	Split VCF	0.000395088	
	3	1_3	vep_{label}_2.tsv vep_{label}_2_header.txt	vep_{label}_3.tsv	Add MSC and GDI	0.000375657	
2	4_1	5_1	vep_{label}_3.tsv	vep_{label}_4_1.tsv biomart_out.txt	Protein-level annotation	0.00792119	
	4_2	8	vep_{label}_3.tsv Gene-level.csv; denovo.txt; RVIS_v3_12Mar16.txt; Gerstein.TXT	vep_{label}_4_2.csv	Gene-level features based on gene name	0.031088888	
	4_3	10	vep_{label}_3.tsv	vep_{label}_4_3.csv	Preprocessing VEP features	0.001683981	
	4_4	11	vep_{label}_3.tsv; ARCHS4_Tissues-2.txt	vep_{label}_4_4.csv	Mapping gene-level binary features from Enrichr libraries.	0.036354569	
3	5	7_1	vep_{label}_4_1.csv	ENSP FASTA files and ENSP out files	netsurfp	0.001042773	Multiple nodes
	6	7_2	vep_{label}_4_1.csv	Prediction output files(IUPRED2 and ANCHOR2)	iupred	0.055623203	Multiple nodes
	7	7_3	vep_{label}_4_1.csv	vep_{label}_7.out	Reading netsurfp results	0.007752792	Multiple nodes
	8	7_4	vep_{label}_7.out	vep_{label}_8.csv	Reading iupred results	0.005168528	Multiple nodes
4	BM*	5_2		biomart_out.txt; mart_out.csv	Getting biomart data		Outside of the pipeline

5	9	6	vep_{label}_4_1.csv mart_out.csv	vep_{label}_9.csv	Creating a column for Pfam domain	1.196721418	
6	10	9	vep_{label}_4_1.csv Phosphorylation.txt; Acetylation.txt; Methylation.txt; Ubiquitination.txt; N-linkedGlycosylation.txt; O-linkedGlycosylation.txt	vep_{label}_10.csv	Mapping the PTM values based on protein name and position	0.040616337	Maximum memory and nodes use
7	11	12	vep_{label}_4_2.csv vep_{label}_4_3.csv vep_{label}_4_4.csv vep_{label}_8.csv vep_{label}_9.csv vep_{label}_10.csv	11_fix.csv	Merging annotations	0.002117931	
	12	13	11_fix.csv; blacklist_hg38.txt;	vep_{label}_11.csv 12.csv	Marking the blacklist	0.00034975	
	13	14	12.csv; vep_{label}_12.csv; LoGoFuncVotingEnsemble_preds_final.csv	vep_{label}_12.csv vep_{label}_13.csv	GOF/LOF/NEU variant	0.003342055	
	14	NA	vep_{label}_13.csv	vep_{label}_14.csv	Standard Formatting		
P	15	NA	vep_{label}_14.csv	vep_{label}_15.csv	Dropping the first column which is empty.		
	16	NA	vep_{label}_15.csv	vep_{label}_16.csv	Dropping and rearranging the columns (please use the column info excel file)		Refer to the column information excel file**
	After 17	NA	vep_{label}_16.csv	vep_{label}_17.csv	Other supplementary modifications upon the user's purpose(s)		
						1.381888132 (In Total)	
	1~16			All the output files between 1 and 15.	Automated pipeline		submit the script, "vep_all_the_way.lsf"

<Stage M (Merging the results)> 1. This stage is required once the pipeline has started from Stage 0. After completing all the stages, come back to the initial working directory and execute STEPM.lsf.							
M	M	NA	vep_{label}_000_16.csv vep_{label}_001_16.csv vep_{label}_002_16.csv vep_{label}_003_16.csv . . . vep_{label}_###_16.csv	vep_{label}_final.csv	The script creates a folder, "STEPM_out in the working directory.	the runtime differs from the number of the files splitted	submit the script, "STEPM.lsf

BM* : BioMart

Column information excel file link:

https://docs.google.com/spreadsheets/d/1m_d_3SAmBzjNek1-SkTt-iuAqBfWaknH/edit?usp=sharing&ouid=102546441492054213912&rtpof=true&sd=true

8. VEP annotation pipeline workflow

<Stage 0>

Stage 0 was devised to reduce the runtime of the pipeline and simultaneously avoid a long pause of the pipeline caused by an extreme workload based on an excessive amount of variants contained in the input vcf. Stage 0 is composed of the total of 4 pSTEPs (preSTEPs): pSTEP1 ~ pSTEP4. pSTEP0 is not included in this stepwise work. However, it is required to input the essential information: 1. working directory; 2. input vcf name; 3. the project label before executing the initiator: VEP_ALL_IN_ONE.lsf or to execute each pre-steps individually. pSTEP1 was devised to split a huge vcf file into smaller pieces of vcf files. The default amount of variants number of the results will be 100,000. This was determined as a result of a runtime calculation experiment which was found to be a comparatively reasonable amount to reduce the workload of HPC server and runtime at the same time. However, the user is allowed to change the amount of variant by modifying the number after the flag, '-l'. After splitting the vcf file, pSTEP1 creates the folders for the splitted vcf files and copies the vcf files into their folders. The name of the vcf files(except the extension, '.vcf') and the folders identical as {label}_###. The role of pSTEP2 is to create STEP0.sh files in each folder which will be used as the essential information including the directory for the outputs, the name of the splitted vcf, and the project label to execute the downstream pipelines. pSTEP3 performs to create the initiator of the VEP annotation pipeline. By taking this step, the initiator named as 'vep_{label}_###.lsf' will come out in each folder, '{label}_###'. Finally pSTEP4 will play a role in the initiator of the initiator. Specifically, once it's executed, All the initiators in all the folders will be executed.

<Stage 1 ~ 7 and Post(P)>

The newly modified VEP annotation pipeline consists of a total of 16 steps (Except the step 0 where the user should input initial information). The entire pipeline was modularized into 7 stages. Each stage contains a different number of steps (number of steps included): Stage 1 (4), Stage 2(4), Stage 3(4), Stage 4(1), Stage

5(1), Stage 6(1), and Stage 7(4). This pipeline was designed to reduce the total running time by executing multiple steps of work at a single stage. Specifically, at stage 2, the total of 4 different steps (4_1, 4_2, 4_3, and 4_4) are executed simultaneously. None of the stages overlap while being run. The purpose of such a design was to meet the maximum use of memory and the number of nodes. Once run, the pipeline follows the stage order from 1 to 7(Except Stage 4. This stage is outside of the pipeline. This step includes the preparation of protein information from Biomart (https://www.ensembl.org/biomart/martview/7ff1e4e6cd72165446874cc42a3b2fd2). Because the input file for stage 5 was already prepared before executing the pipeline, the running time of stage 4 wasn't taken into consideration). In the stages where the multiple steps are involved, after the completion of the last step, the pipeline moves onto the next stage. Exceptionally, in stage 2, after the completion of STEP 4_4 which takes longer time to run than the other steps, the pipeline moves onto the next stage. The total running time of each stage is equivalent to the sum of the running time of the steps included except the stage 2. The total running time of stage 2 is identical to the running time of STEP 4_4 alone because all the other steps within stage 2 end earlier than the STEP 4_4.

<Stage Merge(M)>

This stage was designed to merge all the results which are obtained from Stage 0 (a large input vcf). This step is taken through executing the script, 'STEPM.lsf' in the working directory. Then, the script will create a new folder, 'STEPM_out' and a file, 'vep_{label}_final.csv' within it.

9. Running time Calculation

The multiple steps within the pipeline were automated with the use of "Dependent LSF job submission" where the shell scripts of a step are executed upon the completion of the previous step. Because every single step was executed throughout LSF job submission on HPC(Minerva), the exact start and end time was measured and all the running times of the steps were reported from the HPC accurately. The sample vcf file which was used as the initial input of the pipeline contains the total of 154,396 variants (SNPs). Thus, each running time of every single step was divided by the number of variants. The total running time of the pipeline was calculated as 1.35932926 seconds.
The running time according to either steps or stages as follows:

Table 2. Runtime per step of VEP annotation pipeline

Stage	STEP	Description	Time(per variant) (sec)*
1	0	Input initial information	
	1	HGMD 38 Annotation	0.00623
	2	Split VCF	0.00654
	3	Add MSC and GDI	0.00686
2	4_1	Protein-level annotation	0.00803
	4_2	Gene-level features based on gene name	0.01527
	4_3	Preprocessing VEP features	0.0074
	4_4	Mapping gene-level binary features from Enrichr libraries.	0.01747

3	5	netsurfp	0.0082
	6	iupred	0.01754
	7	Reading netsurfp results	0.01881
	8	Reading iupred results	0.02008
4	BioMart(BM)	Getting biomart data	-
5	9	Creating a column for Pfam domain	0.26219
6	10	Mapping the PTM values based on protein name and position	0.26725
7	11	Merging annotations	0.2692
	12	Marking the blacklist	0.26968
	13	GOF/LOF/NEU variant	0.27479
	14	Standard Formatting	2.74889
Post STEPS	15	Dropping the first column which is empty.	0.00603
	16	Dropping and rearranging the columns (please use the column info excel file)	0.00607
Total			4.23653
Post STEPS	# after 17	Other supplementary modifications upon the user's purpose(s)	-

*Time (per variant) = Run time of 100,000 variants / 100,000 (sec)

10. The relationship between the running time and the number of variants

To ascertain whether there is a linear relationship between the running time and the number of variants and to find the optimum number of nodes and the memory request which are essential to boost the speed while running the pipeline, further computational experiments had been followed. Most of all, in order to compare and find some trends of runtime according to the amount of variants, the pipeline had been independently applied to several different vcf files of different variant sizes: 3, 5, 10, 100, 1000, 10000, and 100000. All of them were derived from one of the versions of vcf from Human Genetic Mutation Database (HGMD_2021.2) to which Genome Reference Consortium 38 (GRCh38) was used as the reference. In addition, before submitting the jobs to the HPC:Minerva, the number of nodes and the amount of memory requested were synchronized step by step (Table 3.). As shown on Table 4., a positive correlation between running time and the number of variants were observed within each step between the variant size of 10 and that of 100,000. However, the small number of variants less than or equal to 5 didn't show a positive correlation against the running time. Thus, the experimental results obtained through the size of 3 and 5 were excluded before performing linear regression modeling. As shown on Table 4, both of the groups showed a trend of taking longer time than the groups of a larger number of variants throughout overall steps. Linear regression test was followed using the software, R (version of 4.1.0) and it has been discovered that there is a strong correlation between running time and the number of variants (Adjusted R-squared = 0.9983) (Fig 4.). After performing logarithmic transformation of both runtime and the number of variants, a strong positive correlation between them has been maintained (Adjusted R-squared: 0.6585).

11. The optimum node(s) numbers, memory request, and walltime(hour) request

To reduce the runtime of the pipeline, it's essential to find the optimum number of node(s) and the amount of memory requested when it comes to submitting the jobs (using bsub). Even though the optimum numbers haven't been obtained due to server changing server conditions and workload, some numbers having shown relatively better performance compared to others were found. Those numbers were introduced as a form of a table across the entire steps of the pipeline. While conducting the calculations of the pipeline, the waltime of 48 hours was used in order to control the condition regardless of the amount of variants. It's been calculated that it takes about 2 hours to complete the annotation of 100,000 variants. Thus, the walltime should be calibrated according to the total number of variants targeted to be annotated. Table 4 shows the amount of runtime used stepwisely while executing the pipeline. Aforementioned, the number of node(s), the amount of memory requested and the walltime requested were controlled the same regardless of the amount of variants (Table 3.). For runtime calculations, small numbers of variants such as 3 and 5 didn't show clear association compared to the others. Hence, the results obtained from those two variants have been removed for the further analyses: bar-graphing and linear regression modeling (Fig 3 ~5.)

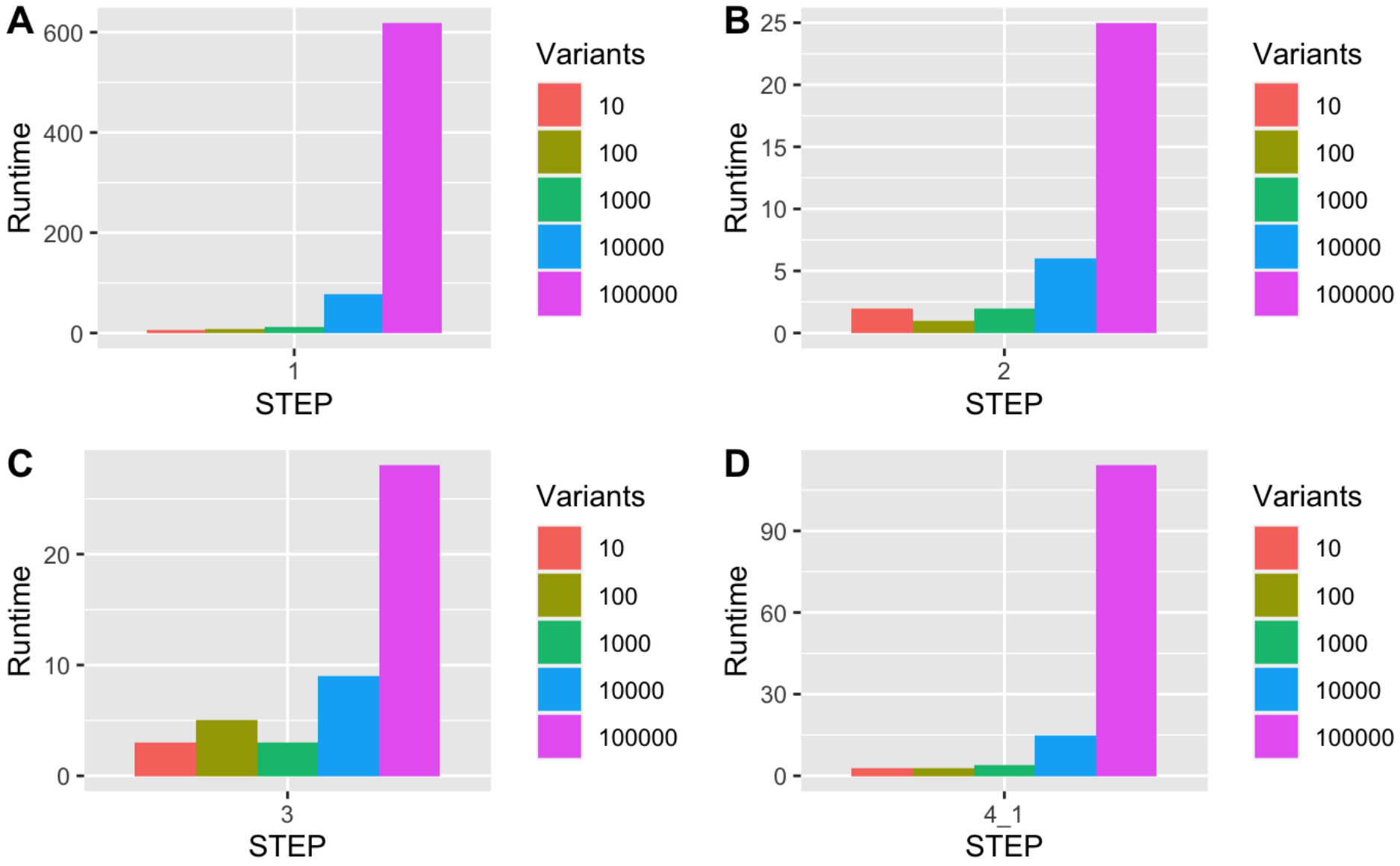
Table 3. Stepwisely requested node(s) number(unit), memory(megabyte), and walltime (hour)

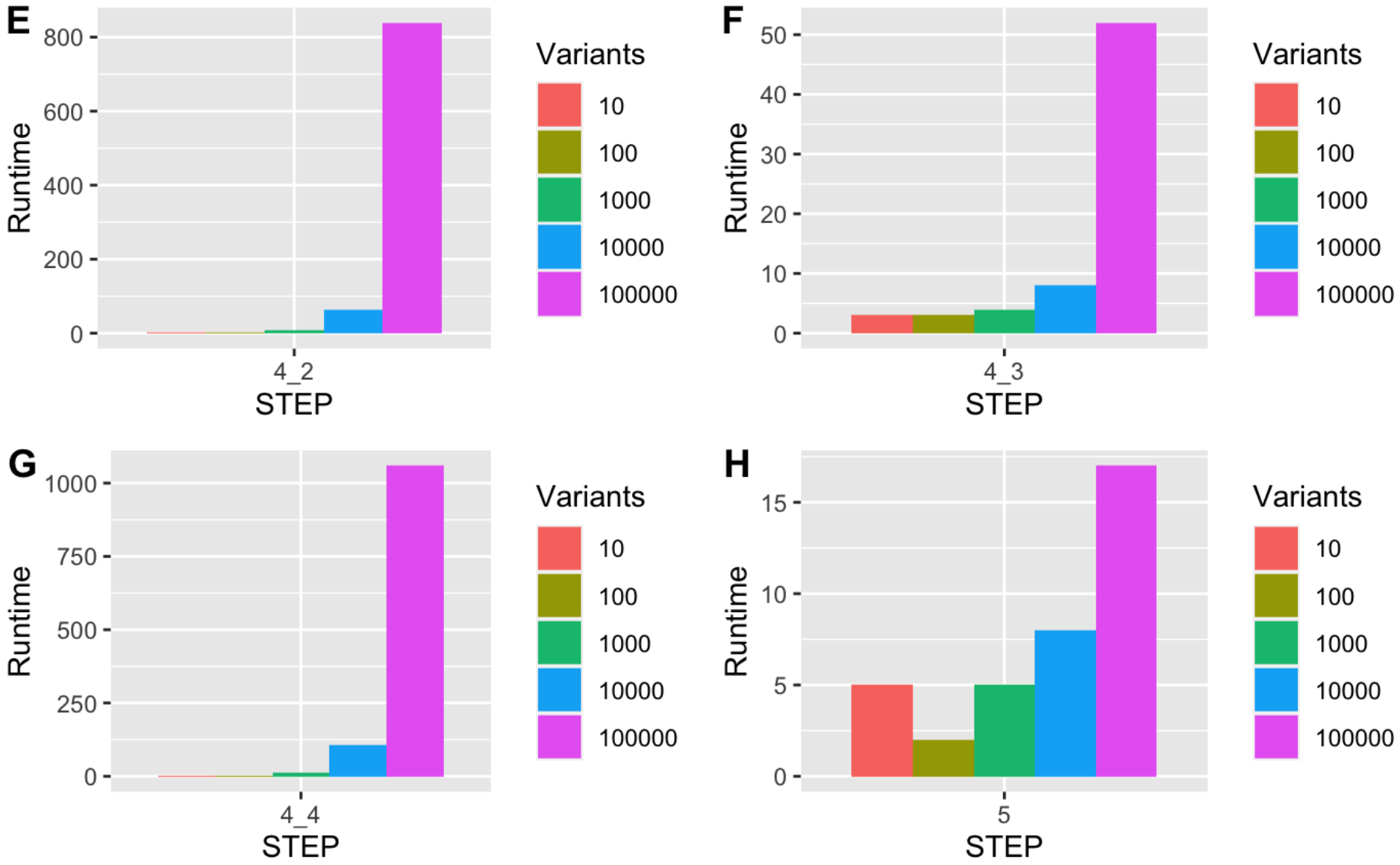
STEP	***1***	***2***	***3***	***4_1***	***4_2***	***4_3***	***4_4***	***5***	***6***	***7***	***8***	***9***	***10***	***11***	***12***	***13***	***14***	***15***	***16***
Node	47	20	20	11	11	11	11	20	20	20	20	20	20	20	20	20	1	20	1

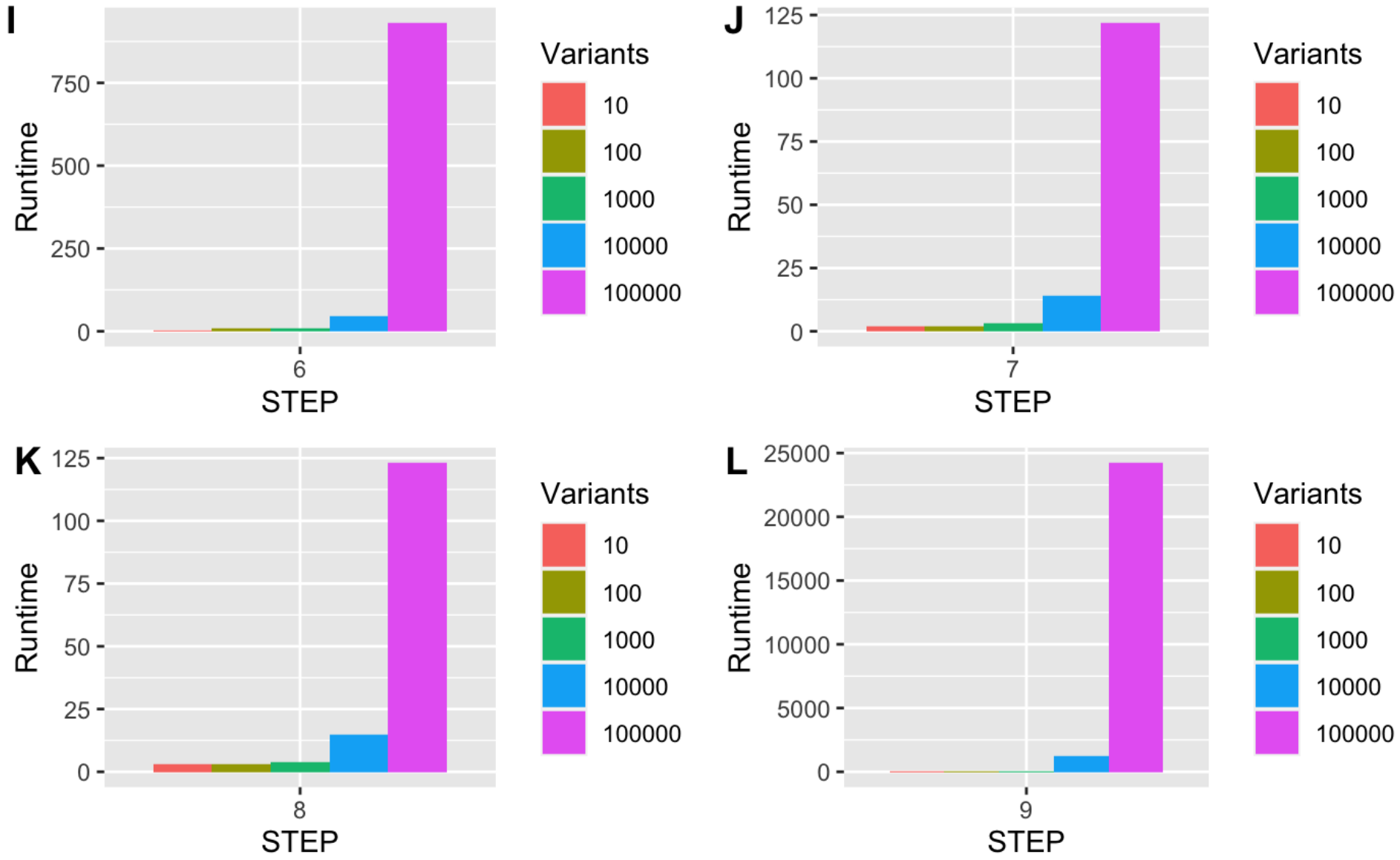
Memory	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	10,000	40,000	10,000	10,000	10,000
Walltime	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48	48

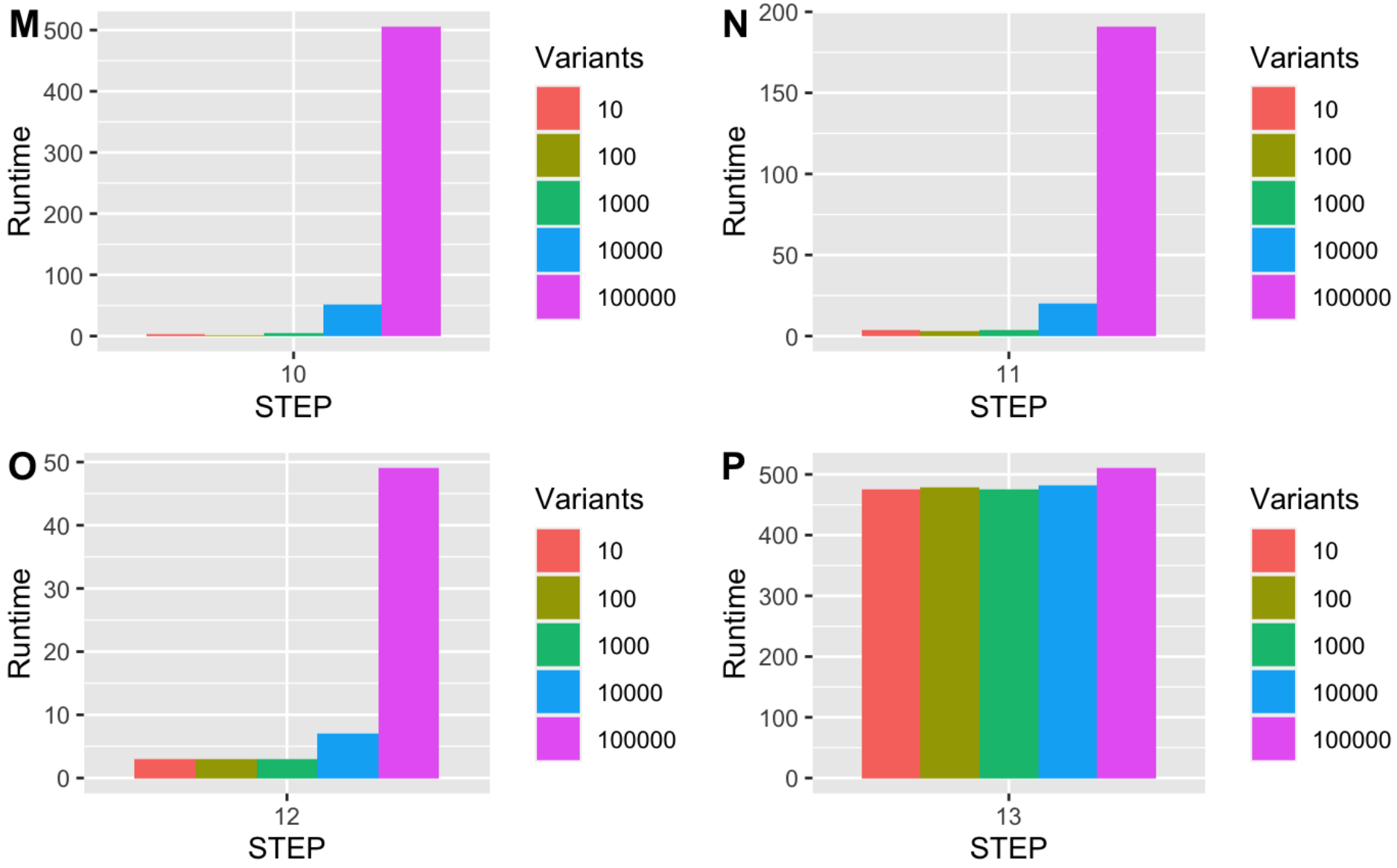
Table 4. Runtime according to the number of variants (second)

# of Variants	*STEP1*	*STEP2*	*STEP3*	*STEP 4_1*	*STEP 4_2*	*STEP 4_3*	*STEP 4_4*	*STEP5*	*STEP6*	*STEP7*	*STEP8*	*STEP9*	*STEP10*	*STEP11*	*STEP12*	*STEP13*	*STEP14*	*STEP15*	*STEP16*	***Total***
3	450	458	463	466	467	472	476	474	481	486	491	505	511	515	520	998	1,004	1,016	1,018	11,271
5	20	26	32	36	37	43	46	48	51	56	61	88	91	95	100	569	574	587	589	3,149
10	11	17	22	28	28	27	28	34	41	47	51	84	91	96	101	579	580	593	595	3,053
100	13	15	24	33	33	33	33	45	50	56	61	95	100	106	110	593	597	604	608	3,209
1,000	30	51	70	91	94	90	97	111	131	146	164	78	273	288	304	793	804	820	829	5,264
10,000	83	89	104	119	168	111	211	126	175	193	208	1,455	1,510	1,534	1,545	2,031	2,034	2,043	2,048	15,787
100,000	623	654	686	803	1,527	740	1747	820	1754	1,881	2,008	26,219	26,725	26,920	26,968	27,479	274,889	603	607	423,653


A
600
400
200
0
Runtime
1
STEP
Variants
10
100
1000
10000
100000
B
25
20
15
10
5
0
Runtime
2
STEP
Variants
10
100
1000
10000
100000
C
20
10
0
Runtime
3
STEP
Variants
10
100
1000
10000
100000
D
90
60
30
0
Runtime
4_1
STEP
Variants
10
100
1000
10000
100000


E
Runtime
800
600
400
200
0
4_2
STEP
Variants
10
100
1000
10000
100000
F
Runtime
50
40
30
20
10
0
4_3
STEP
Variants
10
100
1000
10000
100000
G
Runtime
1000
750
500
250
0
4_4
STEP
Variants
10
100
1000
10000
100000
H
Runtime
15
10
5
0
5
STEP
Variants
10
100
1000
10000
100000


I
Runtime
750
500
250
0
6
STEP
Variants
10
100
1000
10000
100000
J
125
100
75
50
25
0
Runtime
7
STEP
Variants
10
100
1000
10000
100000
K
125
100
75
50
25
0
Runtime
8
STEP
Variants
10
100
1000
10000
100000
L
25000
20000
15000
10000
5000
0
Runtime
9
STEP
Variants
10
100
1000
10000
100000


M
Runtime
500
400
300
200
100
0
10
STEP
Variants
10
100
1000
10000
100000
N
Runtime
200
150
100
50
0
11
STEP
Variants
10
100
1000
10000
100000
O
Runtime
50
40
30
20
10
0
12
STEP
Variants
10
100
1000
10000
100000
P
Runtime
500
400
300
200
100
0
13
STEP
Variants
10
100
1000
10000
100000

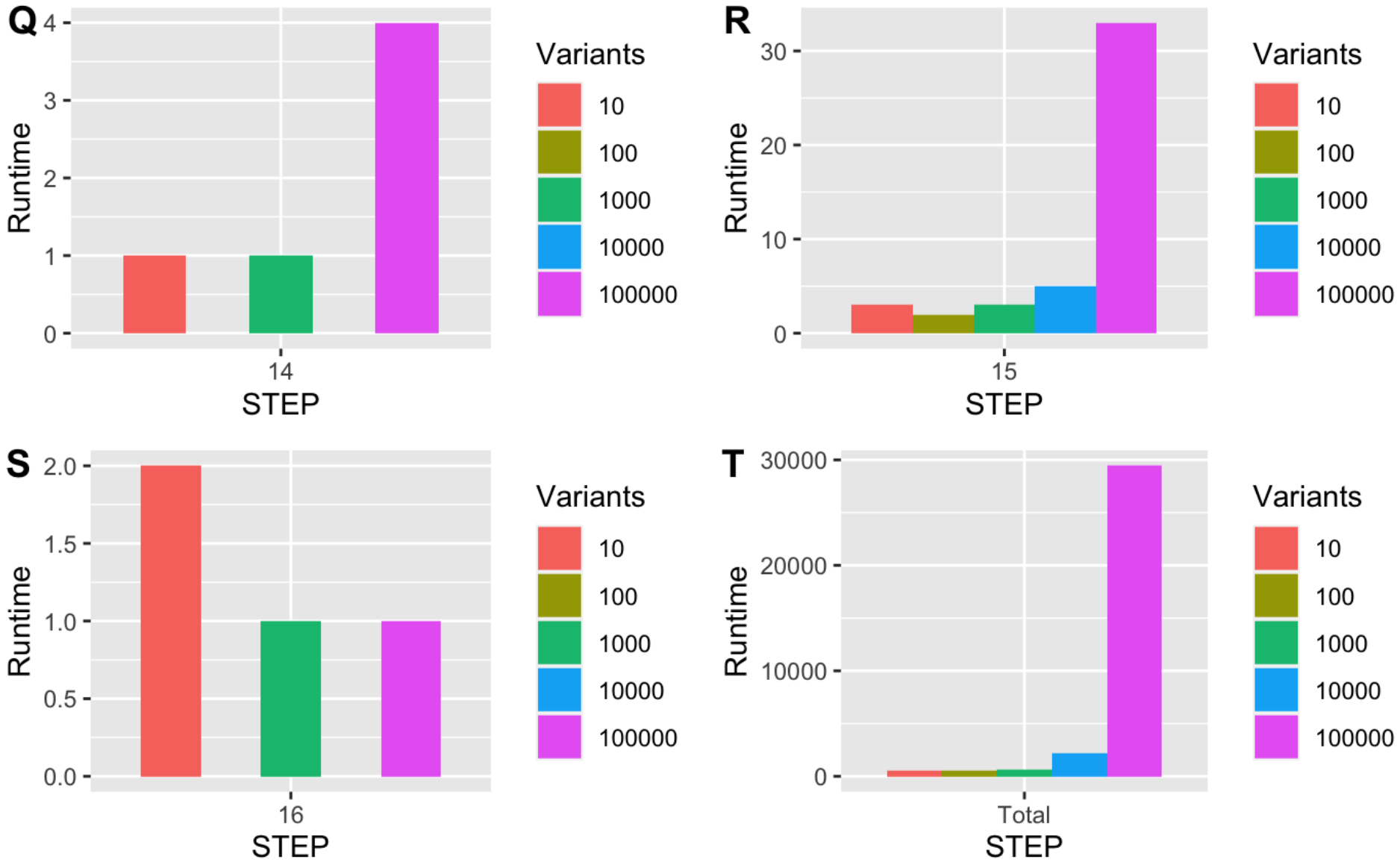


Fig 3. Runtime per step according to different number of variants (A. STEP1, B. STEP2, C. STEP3, D. STEP4_1, E. STEP4_2, F. STEP4_3, G. STEP4_4, H.STEP5, I. STEP6, J. STEP7, K. STEP8, L. STEP9, M. STEP10, N. STEP11, O. STEP12, P. STEP13, Q. STEP14, R. STEP15, S. STEP16, T. Entire STEPs,)

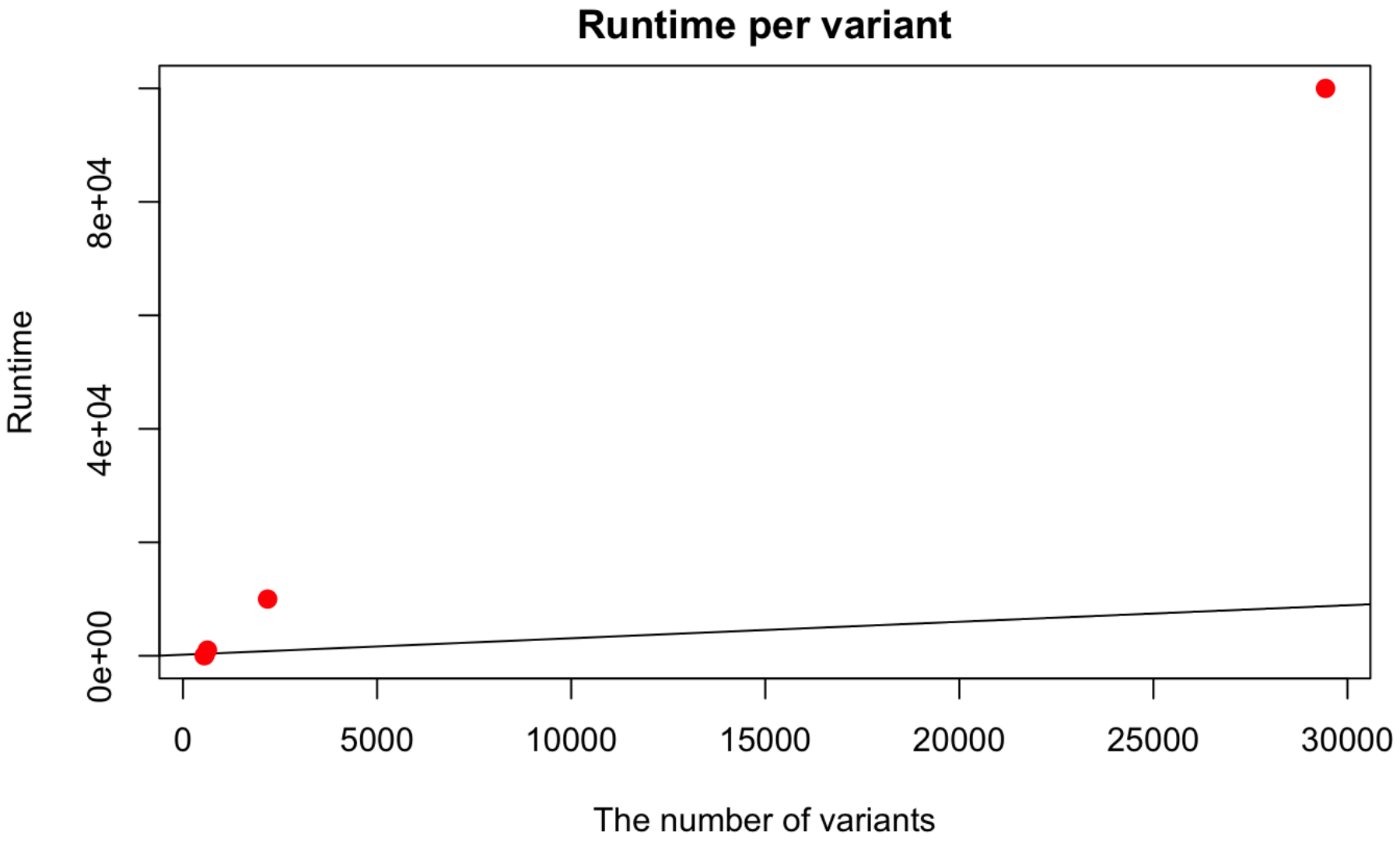


Fig 4. Linear relationship between runtime and the number of variants (Adjusted R-squared = 0.9977)

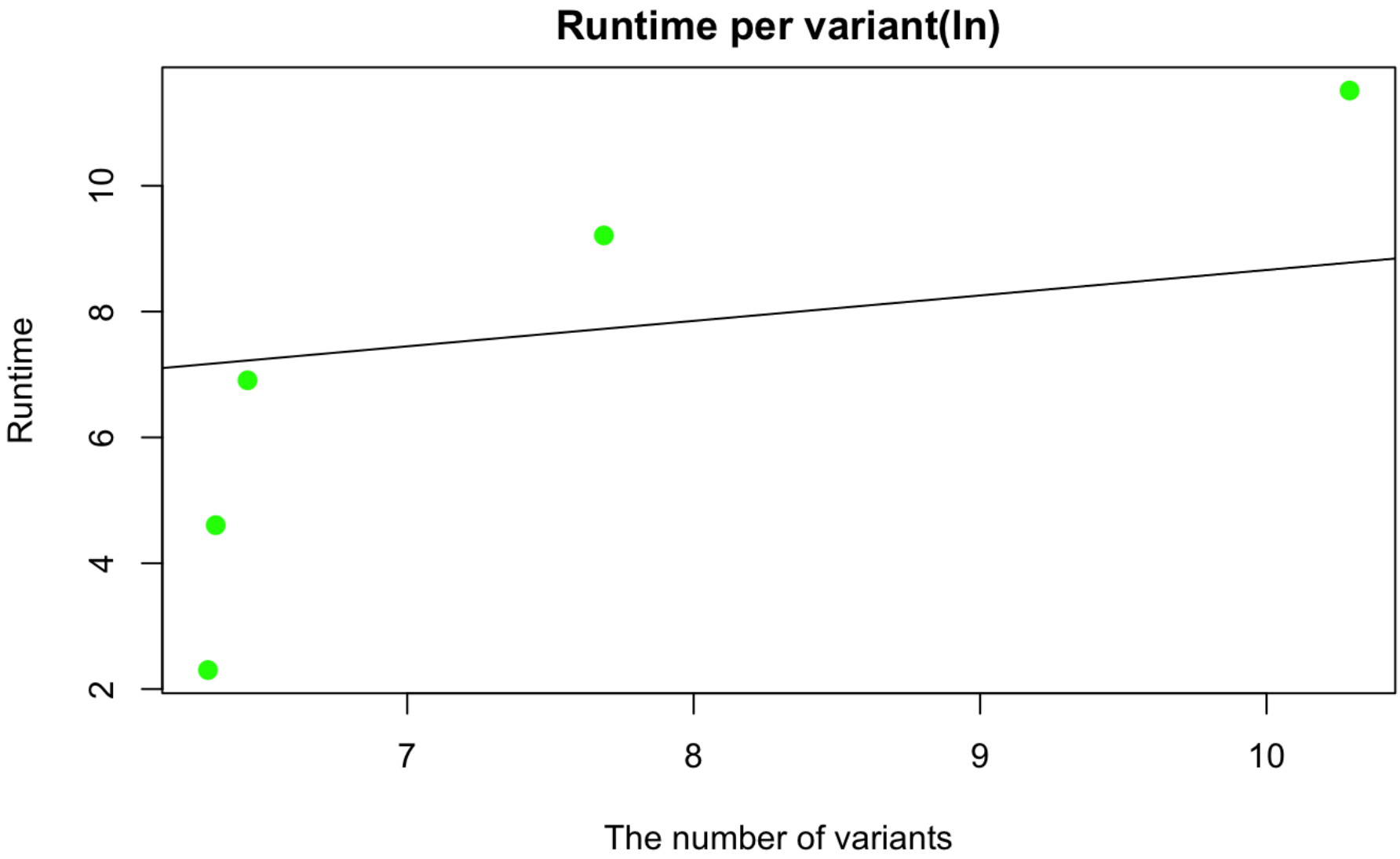


Fig 5. Linear relationship between runtime and the number of variants (log-transformed) (Adjusted R-squared = 0.6585)