Benchmarking Hayai-Annotation Plants: A Re-evaluation Using Standard Evaluation Metrics

Andrea Ghelfi,*[1,2] Kenta Shirasawa,[2] Sachiko Isobe*[2]

[1]Bioinformation and DDBJ Center, National Institute of Genetics, Mishima 411-8540, Japan
[2]Kazusa DNA Research Institute, Kisarazu 292-0818, Japan

*Corresponding Author:
Andrea Ghelfi, E-mail: andreaghelfi@nig.ac.jp
Sachiko Isobe, E-mail: sisobe@kazusa.or.jp

*To whom correspondence should be addressed: andreaghelfi@nig.ac.jp, sisobe@kazusa.or.jp.

Supplementary Text S1. Description of the methodology of "Benchmarking Hayai-Annotation Plants: A Re-evaluation Using Standard Evaluation Metrics".

1. Methodology

1.1 GO annotation evaluations

The genes of *A. thaliana* were randomly selected from a list of all annotated genes. It was considered all unique genes annotated by all Hayai-Annotation, Blast2GO and TRAPID.

```
$ cat GO_blast2go.csv trapid_GO.csv GO_hayai.csv | cut -f2 -d";" | sort | uniq > ara_genes.txt
```

The sampling was performed using the function 'randomNumbers' from the R-package 'random'. The script was run three times, one for each replication.

$ R
```
library(random)
ara <- read.table("ara_genes.txt", header=F)
len <- length(ara$V1)
ran <- randomNumbers(n=1000, min=1, max=len, col=1, base=10, check=TRUE)
nran <- as.data.frame(ran)
nran <- as.numeric(nran$V1)
sample_ara <- ara[nran,]
sample_ara <- sub("T","t",sample_ara)
sample_ara <- sub("G","g",sample_ara)
write.table(sample_ara, "sample_ara_lowercase1.txt", row.names=F, quote=F, col.names=F)
```

The complete directly ontology were added to the gold standard annotation of *A. thaliana* (filename goa_arabidopsis.gaf; downloaded on July 2019; URL, "http://current.geneontology.org/products/pages/downloads.html"), using the file go-basic (downloaded on July, 2019). A file with *A. thaliana* gold standard GO annotation, including the directly ontology was created for each sample.

$ sh gold.sh
```
#!/bin/bash
cut -f5,9,11 ../goa_arabidopsis.gaf > ../temp_benchmark.txt
while IFS= read -r line; do
fgrep -i "$line" ../temp_benchmark.txt >> temp_ara_gold_standards.txt
while IFS= read -r line; do
awk -F"\t" -v var1="$line" -v OFS="\t" '$0 ~ var1 {print var1, $1, $2}' temp_ara_gold_standards.txt >> ara_gold_standards_sample1.txt
done < sample_ara_lowercase1.txt

# Prepare GO files for each main domain.
awk '{if($1 == "id:") {id = $2} ; if($1 == "is_a:") {print id ";" $2}}' go-basic_2019Jul.obo > ontology.txt
awk '{if($1 == "id:") {id = $2} ; if($1 == "namespace:") {print id ";" $2}}' go-basic_2019Jul.obo > namespace.txt
egrep "cellular_component" namespace.txt > obo_cc.txt
egrep "biological_process" namespace.txt > obo_bp.txt
egrep "molecular_function" namespace.txt > obo_mf.txt
```

An Rscript was developed to build the ontology for each GO term in the sample file.

Rscript aragold.R
```
ara_gold <- function() {
  gold <- read.table("ara_gold_standards_sample1.txt", header=F, sep="\t")
  colnames(gold) <- c("gene","go_id","domain")
  ontology <- read.table("../ontology.txt", header=F, sep=";")
  colnames(ontology) <- c("go_id","is_a")
  gold_ontology <- merge(gold, ontology, by="go_id", all.x=T)
  colnames(gold_ontology)[colnames(gold_ontology) == "go_id"] <- "leaf_node"
  colnames(gold_ontology)[colnames(gold_ontology) == "is_a"] <- "go_id"
  k <- 16
  for(i in 1:k){
    gold_ontology <- merge(gold_ontology, ontology, by="go_id", all.x=T)
    name <- paste("P", i, sep="_")
    colnames(gold_ontology)[colnames(gold_ontology) == "go_id"] <- name
    colnames(gold_ontology)[colnames(gold_ontology) == "is_a"] <- "go_id"
  }
  index <- k + 1
  name <- paste("P", index, sep="_")
  colnames(gold_ontology)[colnames(gold_ontology) == "go_id"] <- name
  cols <- dim(gold_ontology)[2]
  gold_ontology <- gold_ontology[, c(cols, 1:(cols-1))]
  gold_ontology <- gold_ontology[, c(cols, (cols-1), 1:(cols-2))]
  write.table(gold_ontology, "sample_gold_ontology.txt", row.names=F, quote=F, col.names=T)
  bp <- gold_ontology[gold_ontology$domain == "P",]
  mf <- gold_ontology[gold_ontology$domain == "F",]
  cc <- gold_ontology[gold_ontology$domain == "C",]
  write.table(bp, "sample_gold_ontology_bp.txt", row.names=F, quote=F, col.names=F)
  write.table(mf, "sample_gold_ontology_mf.txt", row.names=F, quote=F, col.names=F)
  write.table(cc, "sample_gold_ontology_cc.txt", row.names=F, quote=F, col.names=F)
}
ara_gold()
```

A shell script was developed to call the Rscript then prepare the files for AUC-PR calculations.

$ sh prep_files.sh

```
#!/bin/bash
sh gold.sh
Rscript aragold.R
awk '{gsub("NA ", ""); $1=""; print}' sample_gold_ontology_bp.txt | cut -c2- | sort | uniq > clean_sample_gold_ontology_bp.txt
awk '{gsub("NA ", ""); $1=""; print}' sample_gold_ontology_mf.txt | cut -c2- | sort | uniq > clean_sample_gold_ontology_mf.txt
awk '{gsub("NA ", ""); $1=""; print}' sample_gold_ontology_cc.txt | cut -c2- | sort | uniq > clean_sample_gold_ontology_cc.txt
awk '{gsub("NA ", ""); print}' sample_gold_ontology.txt | sort | uniq > all_sample_gold_ontology.txt
awk '$1 == "C"{print}' all_sample_gold_ontology.txt | cut -f3- -d" " > all_sample_gold_ontology_cc.txt
awk '$1 == "F"{print}' all_sample_gold_ontology.txt | cut -f3- -d" " > all_sample_gold_ontology_mf.txt
awk '$1 == "P"{print}' all_sample_gold_ontology.txt | cut -f3- -d" " > all_sample_gold_ontology_bp.txt
```

The precision-recall curves were performed independently for each GO domain (MF, BP, and CC) using the function 'pr' from the R-package 'PRROC' to calculate the integral of the area under the curve.

Calculation of precision, recall and AUC-PR:

```
# shell script was used to remove blank spaces in the files.
awk '{ sub(" GO:", ";GO:"); print }' ../clean_sample_gold_ontology_mf.txt | awk -F";" '{ gsub(" GO", "\n" $1 " GO"); print
$1, $2 }' | sort | uniq > onecol_sample_gold_ontology_mf.txt

# R
all_obo <- read.table("obo_mf.txt", header = F, sep = ";")
colnames(all_obo) <- c("go_id", "domain")
# hayai-annotation: hit1_seq90_score
hayai <- read.table("GO_hayai.csv", header=F, sep=";")
colnames(hayai) <- c("go_id", "gene")
hayai <- merge(hayai, all_obo, by="go_id")
# blast2go
```

```
blast2go <- read.table("GO_blast2go.csv", header=F, sep=";")
colnames(blast2go) <- c("go_id", "gene")
blast2go <- merge(blast2go, all_obo, by="go_id")
# trapid
trapid <- read.table("trapid_GO.csv", header=F, sep=";")
colnames(trapid) <- c("go_id", "gene")
trapid <- merge(trapid, all_obo, by="go_id")
# merge data to agg_ontol
onecol <- read.table("onecol_sample_gold_ontology_mf.txt", header=F)
colnames(onecol) <- c("gene", "go_id")
onecol$gene <- toupper(onecol$gene)
agg_ontol <- onecol
agg_ontol$ontology <- "gold"
# gene list
genes_list <- as.data.frame(unique(agg_ontol[, "gene"]))
colnames(genes_list) <- "gene"
# trapid
trapid <- merge(trapid, genes_list, by= "gene")
# hayai
hayai <- merge(hayai, genes_list, by= "gene")
# blast2go
blast2go <- merge(blast2go, genes_list, by= "gene")

# Calculation of precision
score <- merge (software, agg_ontol, by = c("gene", "go_id"), all.x = T)
wrong <- score[is.na(score$ontology),]
correct <- score[!is.na(score$ontology),]
wrong$eval <- "N"
correct$eval <- "Y"
wrong$count <- 1
correct$count <- 1
agg_eval_wrong <- aggregate(wrong$count, by = list(gene=wrong$gene,eval_wrong=wrong$eval),FUN = sum)
agg_eval_correct <- aggregate(correct$count, by = list(gene=correct$gene,eval_correct=correct$eval),FUN = sum)
colnames(agg_eval_wrong)[3] <- "count_wrong"
colnames(agg_eval_correct)[3] <- "count_correct"
eval <- merge(agg_eval_wrong, agg_eval_correct, by="gene", all = T)
eval[is.na(eval$count_wrong),]$count_wrong <- 0
eval[is.na(eval$count_correct),]$count_correct <- 0
eval$sum <- eval$count_wrong + eval$count_correct
eval$perc <- eval$count_correct/eval$sum
precision <- eval$perc
precision <- precision[!is.na(precision)]

# Calculation of recall
score2 <- merge (software, agg_ontol, by = c("gene", "go_id"), all.y = T)
missing <- score2[is.na(score2$domain),]
correct <- score2[!is.na(score2$domain),]
missing$eval <- "N"
correct$eval <- "Y"
missing $count <- 1
correct$count <- 1
agg_eval_missing <- aggregate(missing $count, by = list(gene= missing $gene,eval_missing=missing$eval),FUN = sum)
agg_eval_correct <- aggregate(correct$count, by = list(gene=correct$gene,eval_correct=correct$eval),FUN = sum)
colnames(agg_eval_missing)[3] <- "count_missing"
colnames(agg_eval_correct)[3] <- "count_correct"
eval <- merge(agg_eval_missing, agg_eval_correct, by="gene", all = T)
eval[is.na(eval$count_missing),]$count_missing <- 0
eval[is.na(eval$count_correct),]$count_correct <- 0
eval$sum <- eval$count_missing + eval$count_correct
eval$perc <- eval$count_correct/eval$sum
recall <- eval$perc

# Calculation of AUC-PR
library (PRROC)
pr <- pr.curve(scores.class0 = precision, scores.class1 = recall, curve = TRUE)
```

The same procedure was done for each the GO domains BP and CC using the same sample 1. Then the script was used for samples 2 and 3 to obtain the replicates data.

The analysis of variance (ANOVA) with Tukey HSD were used to compare the mean of AUC-PR for each software in each one of the three main GO domains, using the functions aov and TukeyHSD using R. The statistical model and the script are shown below. The results of ANOVA are shown in Table S1.

```
stat <- read.table("aucpr_stats.csv", header=T, sep=",")
stat <- stat[!is.na(stat$rep),]
anova <- aov(aucpr_perc ~ software + go_domain + software * go_domain, data = stat)
summary(anova)
TukeyHSD(anova)
```

Table S1. ANOVA comparing means of AUC-PR for the software Hayai-Annotation, Blast2GO and TRAPID in the three main GO domains (MF, BP and CC).

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
software	2	4742	2370.9	2637.1	<2e-16 ***
go_domain	2	973	486.5	541.1	<2e-16 ***
software:go_domain	4	1583	395.7	440.1	<2e-16 ***
Residuals	18	16	0.9		

Signif. code: 0 ‘***’

1.2 Separation of test versus training data

Same analysis was performed with four different values for the parameter sequence identity in Hayai-Annotation (60, 70, 80 and 90%). Two replicates were considered for the purpose of the ANOVA with Tukey HSD calculations. The results of ANOVA are shown in Table S2.

Table S2. ANOVA comparing the means of AUC-PR for different values of sequence identity in Hayai-nnotation.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
parameter	3	50.66	16.89	27.294	1.21e-05 ***
go_domain	2	78.43	39.22	63.382	4.18e-07 ***
parameter:go_domain	6	34.14	5.69	9.197	0.000653 ***
Residuals	12	7.42	0.62		

Signif. code: 0 ‘***’