Method Article

Computational curation and analysis of publicly available protein sequence data from a single protein family

Kyra Dougherty, Katalin A. Hudak*

Department of Biology, York University, Toronto, Canada

ABSTRACT

The wealth of sequence data available on public databases is increasing at an exponential rate, and while tremendous efforts are being made to make access to these resources easier, these data can be challenging for researchers to reuse because submissions are made from numerous laboratories with different biological objectives, resulting in inconsistent naming conventions and sequence content. Researchers can manually inspect each sequence and curate a dataset by hand but automating some of these steps will reduce this burden. This paper is a step-by-step guide describing how to identify all proteins containing a specific domain with the Conserved Protein Domain Architecture Retrieval Tool, download all associated amino acid sequences from NCBI Entrez, tabulate, and clean the data. I will also describe how to extract the full taxonomic information and computationally predict some physicochemical properties of the proteins based on amino acid sequence. The resulting data are applicable to a wide range of bioinformatic analyses where publicly available data are utilized.

- Step-by-step guide to gathering, cleaning, and parsing data from publicly available databases for computational analysis, plus supplementation of taxonomic data and physicochemical characteristics from sequence data.
- This strategy allows for reuse of existing large-scale publicly available data for different downstream applications to answer novel biological questions.

© 2022 The Author(s). Published by Elsevier B.V.
This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)

Abbreviations: RIP, Ribosome inactivating protein.

* Related research article: K. Dougherty, K.A. Hudak Phylogeny and domain architecture of plant ribosome inactivating proteins Phytochemistry, 202 (2022), pp. 113337, 10.1016/j.phytochem.2022.113337
DOI of original article: 10.1016/j.phytochem.2022.113337
Corresponding author.
E-mail address: hudak@yorku.ca (K.A. Hudak).

https://doi.org/10.1016/j.mex.2022.101846
2215-0161/© 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY-NC-ND license (http://creativecommons.org/licenses/by-nc-nd/4.0/)
1 Identify all protein sequences containing the domain of interest

The example used here was the input data for the analyses described in Dougherty and Hudak [3]. The Conserved Domains section of NCBI (https://www.ncbi.nlm.nih.gov/cdd/) contains a database of protein domains collected from a variety of external databases. Here you can search for your domain of interest; for this example the ribosome inactivating protein (RIP) domain will be used (Fig. 1). When you select your domain of interest you will be redirected to a page which outlines details about the domain, including protein structure, related domain families, and representative sequences (Fig. 2). Under the drop-down window called “Links” select “Architectures” to be redirected to the conserved domain architecture retrieval tool [4,9].

Here you will see a graphical view of all the proteins in NCBI with annotations for your query domain, and any other domains that have been annotated as well; they will be separated into combinations of domains. These results can be filtered by taxonomy from the drop-down menu at the top. Under “Filter by taxonomy” select “NCBI taxonomy tree”, select your taxonomic group of interest (in this case plants), select “Include” at the bottom, and click “Apply” at the top to apply the changes.

To access the amino acid sequence data of the identified proteins, navigate to the domain configuration of interest and click “Lookup sequences in Entrez”. This will redirect you to the search results in the Proteins section of NCBI [7]. Download all sequences by selecting “Send to:” > “File” > “FASTA” > “Create file” in the top right corner. If you are interested in investigating more than one domain configuration, as is the case in this example, go back to the previous page and repeat this process for each domain configuration, then copy and paste the sequences into a single FASTA file. The raw data used in this example are available in Supplementary Data 1.

2 Clean and tabulate data in R

The following code blocks are all in the R programming language and were written in RStudio as a markdown file. This file, along with its accompanying HTML output which includes the results of each intermediate step, is available in Supplementary Data 2 and 3, respectively.
2.1 Load in data

Open RStudio and load the required packages: seqinr [2], Biostrings [6], Peptides [5], tidyverse [8], and taxize [1].

library(seqinr) # Biological Sequences Retrieval and Analysis, CRAN v4.2-8
library(Biostrings) # Efficient manipulation of biological strings, Bioconductor v2.62.0
library(Peptides) # Calculate Indices and Theoretical Physicochemical Properties of Protein Sequences, CRAN v2.4.4
library(tidyverse) # Many useful packages for data manipulation and plotting, CRAN v1.3.1
library(taxize) # Taxonomic Information from Around the Web, CRAN v0.9.99

2.2 Import the FASTA file, convert to table

fasta1 <- readAAStringSet("sequence.fasta", use.names=TRUE)
dataset_fasta1 <- data.frame(names(fasta1), paste(fasta1))
colnames(dataset_fasta1) <- c("Name","Sequence")
# Count how many sequences
print(paste0("Number of sequences before filtering: ", nrow(dataset_fasta1)))
Fig. 2. Screenshot of the results page on the Conserved Domain Architecture Retrieval Tool using ‘pfam00161: RIP’ as the query.

| Name | Sequence |
|------|----------|
| CAA65328.1 antiviral protein [Volkamera aculeata] | MKASLVMTMIFGLGVLMFEFARAQTPIAFHVGATISIYTTFIINT... |
| spIP24478.2RIPS_TRIK RecName: Full=Ribosome-inactivating protein amaranthin [Ambrosia artemisiifolia] | MIRFLVFFLLILTLFTAPVEDGVYFSRLGATSSSYGVSINLRKALP... |
| AAD09240.1 ribosome-inactivating protein amaranthin [Ambrosia artemisiifolia] | MIMLIMITVTKSEACQYRTGELHKEKNSPNYANFLRRLRS... |
| AAL15442.1 anti-viral protein PAP [Phytolacca acinosa] | MKSMLVTVSIWVLLAPIPTSTWAINTIIYVNGSTTISKYATFLDNL... |
| NP_001295744.1 ribosome-inactivating protein cucurmosin-1 [Cucurbita pepo] | MKGGMNLMSIMVAIWFCWSICIFGWSAREIEVCPFSSNQNYKA... |
| BAB83666.1 RAP3 [Oryza sativa] | MVKPAAVLLLLYLPLLAPLTPIGLSRNPPNVPVPTIDTREMDVS... |
| spIPQ3464.1RIPA_PHYAM RecName: Full=Antiviral protein a... | MKMMSMVVVMMLSWLILKPSTWAINITFDVGNAIINKYATF... |
| spIP5662.2RIP1_TRIAN RecName: Full=Type I ribosome-inactivating protein a... | MALSSLFFLAIILGSPTAIGDVFSDLSATKKS5YFSFFTQLRLDALPTQG... |
| AAM89504.1 ribosome-inactivating protein type 1 precursor [... | MAASGTMHRULIFMJUAAAAGQGGLVTQFETLDAVLTNRATYTA... |
| AAL61546.1 ribosome-inactivating protein type 1 precursor [... | MHVHLINHKSFSCSQAMKVLQEGKMKMLVMILAWL... |

Fig. 3. Output of ‘head(dataset_fasta1, n=10)’ from the code block in Section 2.4.
2.3 Filter by character patterns

Most sequences will have flags in the FASTA description line indicating if a sequence is incomplete or low quality; therefore, you can remove these sequences with specific keyword searches. The commands shown below are not exhaustive but instead show some examples of potential keywords that can be used for protein data. Partial sequences can be further filtered by removing sequences that do not start with a methionine. The results of this code are visualized in Fig. 3.

```r
dataset_fasta1 <- dataset_fasta1[!str_detect(dataset_fasta1$Name, "partial"),]
dataset_fasta1 <- dataset_fasta1[!str_detect(dataset_fasta1$Name, "[Cc]hain"),]
dataset_fasta1 <- dataset_fasta1[!str_detect(dataset_fasta1$Name, "fragment"),]
dataset_fasta1 <- dataset_fasta1[!str_detect(dataset_fasta1$Name, "LOW"),]
dataset_fasta1 <- dataset_fasta1[!str_detect(dataset_fasta1$Name, "truncated"),]
dataset_fasta1 <- dataset_fasta1[!str_detect(dataset_fasta1$Name, "protein product"),]
# Select only sequences that start with methionine
dataset_fasta1 <- dataset_fasta1[str_detect(dataset_fasta1$Sequence, "^M"),]
# Remove gaps/stop codons (can cause errors in other programs)
dataset_fasta1$Sequence <- gsub("\-\", "", dataset_fasta1$Sequence)
# Count how many sequences survived this filtering process
print(paste0("Number of sequences after this filtering step: ", nrow(dataset_fasta1)))
# Inspect the table (Table 1)
head(dataset_fasta1, n=10)
```

2.4 Identify missing species instances

Some entries will not have the standard notation for species name, which are surrounded by square brackets. Find sequences without species names within the table, then use the accession number to find the species of origin on NCBI and add them manually to the FASTA file. Then you can reload the updated FASTA file into R and continue to the next step. If the output of the following command is empty, then there are no sequences with missing species names and no action is required.

```r
dataset_fasta1[!str_detect(dataset_fasta1$Name, "\["]),]
```

2.5 Extract species names

Extract the species names by selecting the characters between the square brackets in the ‘Names’ column. It may also be useful to replace or delete ‘special characters’ such as periods and spaces, as they can cause errors for other programs in future analyses.

```r
gene_tax1 <- sub("\[\([^\]\[]\)+\]\[.\]\]","\1", dataset_fasta1$Name)
# Replace the spaces with underscores
gene_tax1 <- gsub("\" \", ",gene_tax1)
# Remove the periods
gene_tax1 <- gsub("\."","",gene_tax1)
# Inspect
head(gene_tax1, n=20)
```
Fig. 4. Output of `head(dataset_fasta1, n=10)` from the code block in Section 2.7.

2.6 Clean gene IDs

Clean gene IDs by removing everything except the accession number. Not all submissions will follow the same naming conventions, but all information about a sequence can be retrieved with the accession number so it is the only piece that is necessary to keep. Again, this code is not exhaustive but merely shows some examples of what can be done; be sure to inspect your sequence names to see what kinds of details you need to consider.

```r
# Remove any lowercase letters plus a vertical bar present before the accession number (eg. sp|P22851)
gene_ID1 <- str_remove(gene_ID1, "^[a-z]+\|"")
# Keep only the accession number, plus the version
# This is denoted by a combination of capital letters and numbers and sometimes underscores, followed by a period then a single number
gene_ID1 <- str_extract(gene_ID1, "^[A-Z0-9_]+\.[0-9]"")
# Optional: remove version number
gene_ID1 <- str_remove(gene_ID1, "\.[0-9]"")
head(gene_ID1, n=20)
```

2.7 Add the accession number and species name to separate columns of the original table

The results of this code are visualized in Fig. 4. The results of this code are visualized in Table 2.

```r
dataset_fasta1$Gene_tax <- gene_tax1
dataset_fasta1$Gene_ID <- gene_ID1
# Remove the old 'Names' column
dataset_fasta1[Gene_ID,]
# Inspect (Table 2)
head(dataset_fasta1, n=10)
```
2.8 Check that there are no empty cells in the table

This command will return no results if all cells contain data. If any results are missing an accession number, you can use the amino acid sequence to search your raw data FASTA file and see if this number is missing or if some part of the code caused it to be lost. Row 41 in this example is missing the accession number (Fig. 5), which corresponds to line 484 of the raw FASTA file (Supplementary Data 1). The accession number provided there lacks the version number, which means that the code used in Section 2.6 above for extracting this information found no match with the expected pattern. Because the accession number is present in the FASTA file, the missing data can be added into the table.

```r
# Check for missing data (Table 3)
dataset_fasta1[is.na(dataset_fasta1),]
# Find out which rows are affected (output to console in this case will be: 41)
which(is.na(dataset_fasta1))
# Add missing accession number
dataset_fasta1$Gene_ID[41] <- "2019502A"
# Check again (should be an empty data frame)
dataset_fasta1[is.na(dataset_fasta1),]
```

2.9 Check for duplicate sequences

Check that there are no duplicate sequences by calculating the pairwise percentage identity of all sequences. This is necessary because there are instances where different researchers submitted the sequence of the same gene to NCBI at different times, but the sequences were not 100% identical. The following code will iterate through each sequence and do a pairwise comparison with every other sequence, tabulate the results, and save the entries with a sequence identity over 99% into a new table.

Note, the speed of this process will greatly vary depending on the number of sequences searched and the computational power allocated to R. The dataset used in this example contained approximately 820 sequences and took several minutes to run. The results of this code are visualized in Fig. 6.

```r
end <- length(dataset_fasta1$Gene_ID)
count <- 1:end
```
2.10 Make a table of the duplicates

The output will be all pairwise comparisons in your dataset, including those between other species and to itself. If you are dealing with multiple species, this may result in the identification of orthologs rather than actual duplicates, so these should be excluded. The results of this code are visualized in Fig. 7, and the csv file saved at this step is available under Supplementary Data 4.

```r
# Remove the entries where the query is the same as the test
table_pairwise_I2 <- table_pairwise_I[table_pairwise_I$gene_id_query != table_pairwise_I$gene_id_test,]
# Remove ones where the query and test are from different species
table_pairwise_I2 <- table_pairwise_I2[table_pairwise_I2$gene_tax_query == table_pairwise_I2$gene_tax_test,]
# Save results to a file, for reference (Supplementary Data 4)
write.csv(table_pairwise_I2, "pairwise_percent_identity_over_99.csv", row.names = FALSE)
# Inspect output (Table 5)
head(table_pairwise_I2, n=5)
```

2.11 Remove duplicates

Remove all test sequences that matched with over 99% similarity between two sequences in the same species. The results of this code are visualized in Fig. 8.

```r
dataset_fasta1 <- dataset_fasta1[!dataset_fasta1$Gene_ID %in% table_pairwise_I2$gene_id_query,]
# See how many sequences survived through to this stage of the filtering process
```
Fig. 8. Output of `head(dataset_fasta1, n=10)` from the code block in Section 2.11.

```r
print(paste0("Number after filtering: ", nrow(dataset_fasta1)))
# Inspect table (Table 6)
head(dataset_fasta1, n=10)
```

2.12 Save cleaned and filtered data as a FASTA file

The file generated from this code is available in Supplementary Data 5.

```r
write.fasta(strsplit(dataset_fasta1$Sequence,""), paste(dataset_fasta1$Gene_ID, dataset_fasta1$Gene_tax, sep = "-"), "filtered_sequences.fasta", open = "w", as.string = F)
```

3 Add physicochemical properties and detailed taxonomic information for each sequence

3.1 Tabulate species representation

If you are working with a large dataset from a variety of species, as is the case in this example, it is useful to tabulate the species and how many proteins are associated with each species. This can be repeated later for any taxonomic level by replacing ‘Gene_tax’ with the column name of the taxonomic level of interest. The results of this code are visualized in Fig. 9.

```r
table_summary <- as.data.frame(table(dataset_fasta1$Gene_tax))
colnames(table_summary) <- c("Species", "Number_of_sequences")
# How many species are represented in this dataset?
print(paste0("Total number of species: ", length(table_summary$Species)))
# Inspect table (Table 7)
head(table_summary, n=10)
```

3.2 Make a table of the full taxonomy of each species based on the NCBI taxonomy classification

Note that this will take several minutes to run as retrieving the data for each species takes a couple of seconds. The results of this code are visualized in Fig. 10, and the csv file generated from this code is available under Supplementary Data 6.

```r
# Convert from data type 'factor' to 'character'
```
### Fig. 9. Output of ‘head(table_summary, n=10)’ from the code block in Section 3.1.

| Species                        | Number of sequences |
|--------------------------------|---------------------|
| Abrus_precatorius              | 19                  |
| Abrus_pulchellus_subsp_tenuiflorus | 4                  |
| Aegilops_tauschii_subsp_strangulata | 2                  |
| Amaranthus_tricolor            | 1                   |
| Amaranthus_viridis             | 2                   |
| Ananas_comosus                 | 3                   |
| Artemisia_annua                | 10                  |
| Benincasa_hispida              | 4                   |
| Beta_vulgaris_subsp_vulgaris    | 22                  |
| Brachypodium_distachyon        | 18                  |

### Fig. 10. Output of ‘head(taxonomy_summary, n=5)’ from the code block in Section 3.2.

```r
table_summary$Species <- as.character(table_summary$Species)
nspecies <- length(table_summary$Species)
# Make empty data frame
full_tax <- data.frame(Species=table_summary$Species,
Genus=character(nspecies), Family=character(nspecies),
Order=character(nspecies), Class=character(nspecies),
Phylum=character(nspecies))
# Fill in data frame for each protein
for (i in full_tax$Species){
  full_tax$Genus[full_tax$Species == i] <- tax_name(i, get = "genus",
  db = "ncbi")$genus
  full_tax$Family[full_tax$Species == i] <- tax_name(i, get = "family",
  db = "ncbi")$family
  full_tax$Order[full_tax$Species == i] <- tax_name(i, get = "order",
  db = "ncbi")$order
  full_tax$Class[full_tax$Species == i] <- tax_name(i, get = "class",
  db = "ncbi")$class
  full_tax$Phylum[full_tax$Species == i] <- tax_name(i, get = "phylum",
  db = "ncbi")$phylum
}
```
3.3 Calculate physicochemical properties

Computationally infer physicochemical properties for each amino acid sequence: aliphatic index, Bowman potential protein interaction index, theoretical net charge, hydrophobicity index, molecular weight, monoisotopic mass over charge ratio, and isoelectric point. This package can calculate more properties than what is shown here, so this is just an example of some of them. Note: if there are unusual characters in your sequence (e.g., B, U, X, Z, *, or any number) then this code will produce an error. You can remove these sequences in the same way you removed those that did not start with a methionine. Alternatively, you can replace the amino acid with another character or with nothing (i.e., empty quotes) the same way as was done to remove special characters from the taxonomic names.

```r
data(dataset_fasta1$aliphatic_index <- aIndex(dataset_fasta1$Sequence)
data(dataset_fasta1$Boman_Potential_Protein_Interaction_index <- boman(dataset_fasta1$Sequence)
data(dataset_fasta1$theoretical_net_charge <- charge(dataset_fasta1$Sequence, pH = 7, pKscale = "Lehninger")
data(dataset_fasta1$hydrophobicity_index <- hydrophobicity(dataset_fasta1$Sequence, scale = "KyteDoolittle")
data(dataset_fasta1$instability_index <- instaIndex(dataset_fasta1$Sequence)
data(dataset_fasta1$molecular_weight <- mw(dataset_fasta1$Sequence)
data(dataset_fasta1$monoisotopic_mass_over_charge_ratio <- mz(dataset_fasta1$Sequence)
data(dataset_fasta1$isolectric_point <- pI(dataset_fasta1$Sequence, pKscale = "EMBOSS")
head(dataset_fasta1[order(dataset_fasta1$Gene_ID)], n = 10)
```

3.4 Combine results into a single table

The csv file generated from this code is available under Supplementary Data 7, and the text file is available under Supplementary Data 8.

```r
for (i in dataset_fasta1$Gene_tax){
data(dataset_fasta1$Genus[dataset_fasta1$Gene_tax == i] <- taxonomy_summary$Genus[taxonomy_summary$Species == i]
data(dataset_fasta1$Family[dataset_fasta1$Gene_tax == i] <- taxonomy_summary$Family[taxonomy_summary$Species == i]
data(dataset_fasta1$Order[dataset_fasta1$Gene_tax == i] <- taxonomy_summary$Order[taxonomy_summary$Species == i]
data(dataset_fasta1$Class[dataset_fasta1$Gene_tax == i] <- taxonomy_summary$Class[taxonomy_summary$Species == i]
data(dataset_fasta1$Phylum[dataset_fasta1$Gene_tax == i] <- taxonomy_summary$Phylum[taxonomy_summary$Species == i])
}
head(dataset_fasta1, n = 10)
```

```r
write.csv(dataset_fasta1, file = "tabulated_cleaned_data.csv", row.names = FALSE)
```
write.table(dataset_fasta1$Gene_ID, ‘‘accessions.txt’’, quote=FALSE, row.names=FALSE, col.names=FALSE)
  
# Inspect
head(dataset_fasta1, n=20)

The final output of this process is included in Dougherty and Hudak [3], Supplementary Data 1 and Supplementary Data 2.

4 Manual inspection of sequences

The protein sequence data on NCBI come from a variety of sources with different experimental purposes. Therefore, it may be necessary to assess the quality of all sequences and filter any that do not meet the standards of your experiment. While many of these steps have been done in R, some manual inspection is still advised by reading the GenPept entries of each sequence. Some useful details available on these pages are 1: whether sequences are genomic in origin or clones from cDNA, and 2: whether sequences are annotated as mature peptides. To view the GenPept pages of only the sequences that passed previous filtering steps you can use Batch Entrez (https://www.ncbi.nlm.nih.gov/sites/batchentrez, [7]). Upload the text file containing the NCBI accession numbers (accessions.txt), select “Protein” and select “Retrieve”. You will be redirected to a page indicating how many records were successfully retrieved. Click the link “Retrieve records”, and you will be redirected again to the Proteins database on NCBI where you can inspect each sequence or download the GenPept files.

5 Method validation

Because each step is performed in RStudio, and because the cleaning and reorganizing of data are done in stages, it is straightforward to inspect the data at each step to ensure that the changes being made are expected. This inspection was done in the case of the example used here; all relevant data were retained and all data from incomplete sequences, low quality sequences, and duplicates were removed. In addition, all irrelevant information from the FASTA description lines was removed and the filtered sequences were successfully written to a new FASTA file and their description lines contained only the NCBI accession number and the species name. These cleaned data have many potential bioinformatic applications; for further detail on the subsequent analyses used with this dataset see Dougherty and Hudak [3].

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

All code and data are available in supplementary materials.

Acknowledgments

FUNDING: This work was supported by a Discovery Grant to K.A.H. from the Natural Sciences and Engineering Research Council of Canada, and a Canada Graduate Scholarship – Master’s (CGS M) to K.D.

Supplementary materials

Supplementary material associated with this article can be found, in the online version, at doi: 10.1016/j.mex.2022.101846.
References

[1] S.A. Chamberlain, E. Szöcs, Taxize: taxonomic search and retrieval in R, F1000 Res. 2 (2013) 191, doi:10.12688/f1000research.2-191.v1.
[2] D. Charif, J.R. Lobry, Seqin[R] 1.0-2: a contributed package to the [R] project for statistical computing devoted to biological sequences retrieval and analysis, Struct. Approaches Seq. Evol. (2007) 207–232, doi:10.1007/978-3-540-35306-5_10.
[3] K. Dougherty, K.A. Hudak, Phylogeny and domain architecture of plant ribosome inactivating proteins, Phytochemistry 202 (2022) 113337, doi:10.1016/j.phytochem.2022.113337.
[4] L.Y. Geer, M. Domrachev, D.J. Lipman, S.H. Bryant, CDART: protein homology by domain architecture, Genome Res. 12 (2002) 1619–1623, doi:10.1101/gr.278202.
[5] D. Osorio, P. Rondon-Villarreal, R. Torres, Peptides: a package for data mining of antimicrobial peptides, R. J. 7 (2015) 4–14, doi:10.32614/RJ-2015-001.
[6] H. Pagès, P. Aboyoun, R. Gentleman, S. DebRoy, Biostrings: efficient manipulation of biological strings R package version 2.62.0. (2021) https://bioconductor.org/packages/Biostrings/.
[7] E.W. Sayers, E.E Bolton, J.R. Brister, K. Canese, J. Chan, D.C. Comeau, R. Connor, K. Funk, C. Kelly, S. Kim, T. Madej, A. Marchler-Bauer, C. Lanczycki, S. Lathrop, Z. Lu, F. Thibaud-Nissen, T. Murphy, L. Phan, Y. Skripchenko, T. Tse, J. Wang, R. Williams, B.W. Trawick, K.D. Pruitt, S.T. Sherry, Database resources of the national center for biotechnology information, Nucleic Acids Res. 50 (2022), doi:10.1093/nar/gkab1112.
[8] H. Wickham, M. Averick, J. Bryan, W. Chang, L. D’Agostino McGowan, R. François, G. Grolemund, A. Hayes, L. Henry, J. Hester, M. Kühn, T.L. Pedersen, E. Miller, S.M. Bache, K. Müller, J. Ooms, D. Robinson, D.P. Seidel, V. Spinu, K. Takahashi, D. Vaughan, C. Wilke, K. Woo, H. Yutani, Welcome to the tidyverse, J. Open Source Softw. 4 (2019) 1686, doi:10.21105/joss.01686.
[9] M. Yang, M.K. Derbyshire, R.A. Yamashita, A. Marchler-Bauer, NCBI’s conserved domain database and tools for protein domain analysis, Curr. Protoc. Bioinform. 69 (2020), doi:10.1002/cpbi.90.