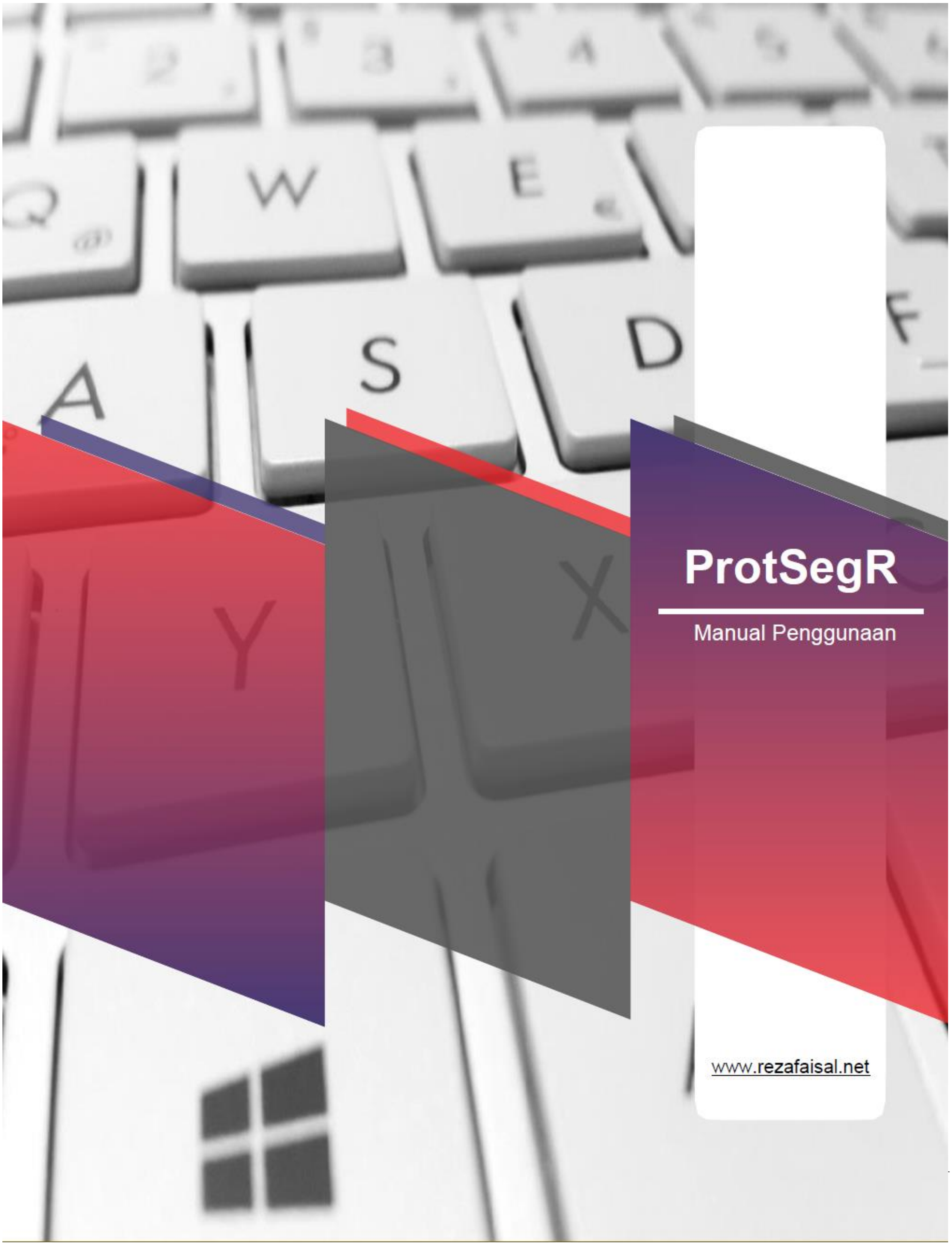


ProtSegR

Manual Penggunaan

www.rezafaisal.net

Daftar Isi

Instalasi Aplikasi Pendukung

R Environtment

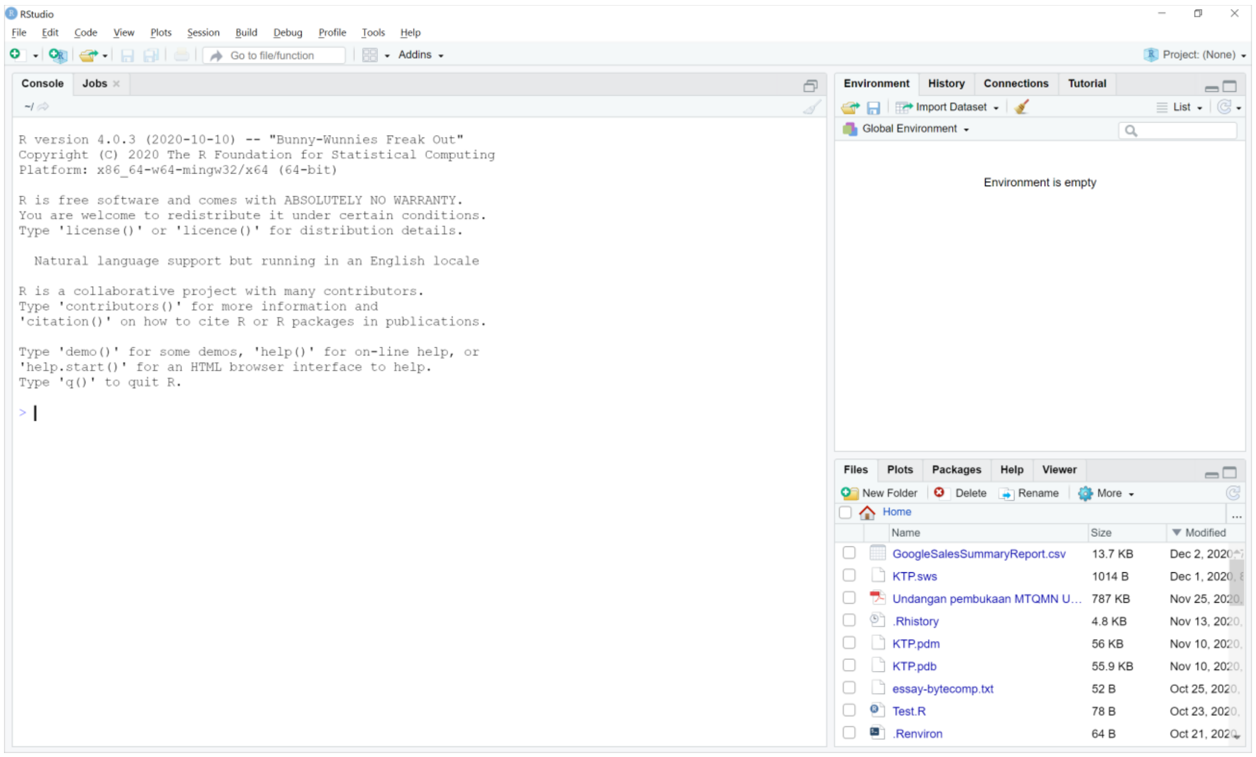
R environtment dapat diunduh pada link berikut: https://cran.r-project.org/bin/windows/base/old/ Pada halaman tersebut pilih versi R enviroment yang ingin diinstall. Pada saat manual ini ditulis, versi terbaru adalah file R-4.0.3-win.exe.

Setelah file R-4.0.3-win.exe diunduh, klik double pada file tersebut kemudian akan ditampilan window panduan installasi. Ikuti langkah-langkah pada windows tersebut sampai selesai.

R Studio

R Studio adalah Tool IDE yang digunakan untuk menulis kode bahasa pemrograman R. Installer R Studio dapat diunduh pada link berikut: https://rstudio.com/products/rstudio/download/. Pada saat manual ini dibuat, versi terbaru R Studio adalah file RStudio-1.3.1093.exe.

Setelah file RStudio-1.3.1093.exe selesai diunduh, klik double pada file tersebut kemudian ikuti langkah-langkah installasi sampai selesai. Setelah proses installasi selesai jalankan R Studio maka dapat dilihat tampilan sebagai berikut.



```
R version 4.0.3 (2020-10-10) -- "Bunny-Wunnies Freak Out"
Copyright (C) 2020 The R Foundation for Statistical Computing
Platform: x86_64-w64-mingw32/x64 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

  Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

> |
```

Name	Size	Modified
GoogleSalesSummaryReport.csv	13.7 KB	Dec 2, 2020
KTP.sws	1014 B	Dec 1, 2020
Undangan pembukaan MTQMN U...	787 KB	Nov 25, 2020
.Rhistory	4.8 KB	Nov 13, 2020
KTP.pdm	56 KB	Nov 10, 2020
KTP.pdb	55.9 KB	Nov 10, 2020
essay-bytecomp.txt	52 B	Oct 25, 2020
Test.R	78 B	Oct 23, 2020
.Renviron	64 B	Oct 21, 2020

Gambar 1. RStudio.

Package devtools

Package ini berfungsi untuk menginstall program utama. Untuk menginstall package ini diperlukan koneksi internet, karena proses installasinya adalah dengan mengunduh package secara langsung saat proses installasi.

Ketikkan perintah ini pada console R Studio yang telah dijalankan seperti pada Gambar 1.

```
install.packages("devtools")
```

Untuk menjalankan atau memuat package ini gunakan perintah berikut.

```
library(devtools)
```

Package Protein Descriptor

Package untuk protein descriptor yang digunakan adalah protr. Untuk menginstall package ini digunakan perintah berikut yang dituliskan pada console pada R Studio.

```
install.packages("protr")
```

Untuk menjalankan atau memuat package ini gunakan perintah berikut.

```
packages(protr)
```

Installasi Program Utama

Program utama adalah package ProtSegR. Package ini disimpan pada GitHub dengan link berikut ini https://github.com/rezafaisal/ProtSegR.

Untuk menginstall package ProtSegR digunakan perintah berikut.

```
library(devtools)
install_github("rezafaisal/ProtSegR")
```

Proses dan hasil installasi adalah sebagai berikut.

```
Downloading GitHub repo rezafaisal/ProtSegR@HEAD
√  checking for file 'C:\Users\M Reza Faisal\AppData\Local\Temp\RtmpC02lgM\r
emotes24942c19185f\rezafaisal-ProtSegR-6cf8f23/DESCRIPTION' (901ms)
-  preparing 'ProtSegR': (747ms)
√  checking DESCRIPTION meta-information ...
-  checking for LF line-endings in source and make files and shell scripts (
509ms)
-  checking for empty or unneeded directories
-  building 'ProtSegR_0.1.13.tar.gz'
```

```
Installing package into 'C:/Users/M Reza Faisal/Documents/R/win-library/4.0'
(as 'lib' is unspecified)
* installing *source* package 'ProtSegR' ...
** using staged installation
** R
** byte-compile and prepare package for lazy loading
** help
*** installing help indices
  converting help for package 'ProtSegR'
    finding HTML links ... done
    convertToFeatureRepresentation          html
    generateSegments                        html
    removeUnrecognizeAminoAcid              html
** building package indices
** testing if installed package can be loaded from temporary location
** testing if installed package can be loaded from final location
** testing if installed package keeps a record of temporary installation pat
h
* DONE (ProtSegR)
```

Source Code Program

Program convertToFeatureRepresentation.R

convertToFeatureRepresentation.R

```
# ConvertToFeatureRepresentation
# ===========================================================================
# This function generate feature representation from original sequence, adjacent and overlapped
segments
#
# You can learn more about package authoring with RStudio at:
#
#   http://r-pkgs.had.co.nz/
#
# Some useful keyboard shortcuts for package authoring:
#
#   Build and Reload Package:  'Ctrl + Shift + B'
#   Check Package:             'Ctrl + Shift + E'
#   Test Package:              'Ctrl + Shift + T'

ConvertToFeatureRepresentation <- function(seq_str, z, prot_desc) {
  if(z < 2){
    stop("z must be bigger than 1.")
  }

  original_sequence = seq_str

  if(exists("main_data")){
    rm("main_data")
  }

  prot_desc_arr = unlist(strsplit(prot_desc, ','))

  #generate input with z value (original sequence, adjacent & overlapped segments)
  #--------------------------------------------------------------------------------- start
  if(exists("data_segments")){
    rm("data_segments")
  }

  for(k in 2:z){
    data_segments_temp = GenerateSegments(seq_str, k)
    rownames(data_segments_temp) = paste0(k, "#", rownames(data_segments_temp))
```

convertToFeatureRepresentation.R

```
    if(!exists("data_segments")){
      assign("data_segments", data_segments_temp)
    } else {
      data_segments = rbind(data_segments, data_segments_temp)
    }

    #print(GenerateSegments(seq_str, k))
    #print("---------------------------")
  }

  #add original sequence to data frame of segments
  data_segments = rbind(original_sequence, data_segments)
  rownames(data_segments)[1] = "1#original"
  data_segments.rownames = rownames(data_segments)
  print(rownames(data_segments))
  #-------------------------------------------------------------------------------- end

  #looping for protein descriptor
  for(prot_desc_i in prot_desc_arr){
    #print(prot_desc_i)

    #generate feature representation
    for(data_input_i in 1:nrow(data_segments)){
      #print(paste(input_i,"=============================="))
      input_i = data_segments[data_input_i,]
      print(data_segments[data_input_i,])
      divider_i = GetKValueFromColName(data_segments.rownames[data_input_i])
      # print("------------")
      # print(data_segments.rownames[data_input_i])
      # print(divider_i)

      result = ConvertToFeatureRepresentationByDescriptor(input_i, prot_desc_i)
      result = t(result)
      colnames(result) = paste0(prot_desc_i, divider_i,".", colnames(result))

      if(!exists("main_data")){
        assign("main_data", result)
      } else {
        main_data = cbind(main_data, result)
      }
    }
  }

  return(main_data)
}
```

Program convertToFeatureRepresentationByDescriptor.R

convertToFeatureRepresentationByDescriptor.R

```
# ConvertToFeatureRepresentation
# =========================================================================
# This function generate feature representation from original sequence, adjacent and overlapped
segments
#
# You can learn more about package authoring with RStudio at:
#
#   http://r-pkgs.had.co.nz/
#
# Some useful keyboard shortcuts for package authoring:
#
#   Build and Reload Package:  'Ctrl + Shift + B'
#   Check Package:             'Ctrl + Shift + E'
#   Test Package:              'Ctrl + Shift + T'


ConvertToFeatureRepresentationByDescriptor <- function(seq_str, prot_desc) {
  library(protr)
```

convertToFeatureRepresentationByDescriptor.R

```
  feature_representation = ""

  if(prot_desc == "AAC"){
    feature_representation = extractAAC(seq_str)
  }
  else if(prot_desc == "DC"){
    feature_representation = extractDC(seq_str)
  }
  else if(prot_desc == "CTDC"){
    feature_representation = extractCTDC(seq_str)
  }
  else if(prot_desc == "CTDT"){
    feature_representation = extractCTDT(seq_str)
  }
  else if(prot_desc == "CTDD"){
    feature_representation = extractCTDD(seq_str)
  }

  return(feature_representation)
}
```

Program generateSegments.R

generateSegments.R

```
# GenerateSegments
# ======================================================================
# This function generate adjacent and overlapped segments from a protein sequence
#
# You can learn more about package authoring with RStudio at:
#
#   http://r-pkgs.had.co.nz/
#
# Some useful keyboard shortcuts for package authoring:
#
#   Build and Reload Package:  'Ctrl + Shift + B'
#   Check Package:             'Ctrl + Shift + E'
#   Test Package:              'Ctrl + Shift + T'

GenerateSegments <- function(seq_str, k, is.original = FALSE) {
  divider_i = k
  seq = RemoveUnrecognizeAminoAcid(seq_str)

  seq.length = nchar(seq)
  part_length = round(seq.length/divider_i)

  if(exists("main_data")){
    rm("main_data")
  }

  #generate adjacent segments
  if(exists("adjacent_segments")){
    rm("adjacent_segments")
  }

  for(j in 1:divider_i){
    if(j > 1){
      start = ((j - 1) * part_length) + 1
      end = j * part_length

      if(j == divider_i){
        end = seq.length
      }
    } else {
```

generateSegments.R

```
      start = 1
      end = part_length
    }

    seq.part = substr(seq, start, end)

    if(!exists("adjacent_segments")){
      assign("adjacent_segments", seq.part)
    } else {
      adjacent_segments = rbind(adjacent_segments, seq.part)
    }
  }

  #generate overlapped segments
  if(exists("overlapped_segments")){
    rm("overlapped_segments")
  }

  for(j in 1:(divider_i-1)){
    half_adj_i    =    substr(adjacent_segments[j,],    (round(nchar(adjacent_segments[j,])/2)),
nchar(adjacent_segments[j,]))
    half_adj_i_plus_1             =             substr(adjacent_segments[j+1,],            0,
(round(nchar(adjacent_segments[j+1,])/2)))

    seq.part = paste0(half_adj_i, half_adj_i_plus_1)

    if(!exists("overlapped_segments")){
      assign("overlapped_segments", seq.part)
    } else {
      overlapped_segments = rbind(overlapped_segments, seq.part)
    }

    if(j == divider_i){
      break()
    }
  }

  #combine segments data
  if(is.original){
    main_data = rbind(seq, adjacent_segments, overlapped_segments)
    rownames(main_data)[1] = "original"
    rownames(main_data)[2:(divider_i+1)] = paste0("adjacent_", 1:divider_i)
    rownames(main_data)[(divider_i+2):(2*divider_i)] = paste0("overlapped_", 1:(divider_i-1))
  } else {
    main_data = rbind(adjacent_segments, overlapped_segments)
    rownames(main_data)[1:(divider_i)] = paste0("adjacent_", 1:divider_i)
    rownames(main_data)[(divider_i+1):(2*divider_i-1)] = paste0("overlapped_", 1:(divider_i-1))
  }

  colnames(main_data)[1] = "amino_acids"

  #return value
  return(main_data)
}
```

Program getKValueFromColName.R

getKValueFromColName.R

```
# GetKValueFromColName
# ========================================================================
# This function generate feature representation from original sequence, adjacent and overlapped
segments
#
# You can learn more about package authoring with RStudio at:
#
#   http://r-pkgs.had.co.nz/
#
```

getKValueFromColName.R

```
# Some useful keyboard shortcuts for package authoring:
#
#   Build and Reload Package:  'Ctrl + Shift + B'
#   Check Package:             'Ctrl + Shift + E'
#   Test Package:              'Ctrl + Shift + T'

GetKValueFromColName <- function(colname_str) {
  colname_str.arr = unlist(strsplit(colname_str, '#'))

  return(colname_str.arr[1])
}
```

Program removeUnrecognizeAminoAcid.R

removeUnrecognizeAminoAcid.R

```
# RemoveUnrecognizeAminoAcid
# ========================================================================
# This function remove amino acid code that is not in these 20 amino acids:
# Glycine/G
# Alanine/A
# Valine/V
# Cysteine/C
# Proline/P
# Leucine/L
# Isoleucine/I
# Methionine/M
# Tryptophan/W
# Phenylalanine/F
# Lysine/K
# Arginine/R
# Histidine/H
# Serine/S
# Threonine/T
# Tyrosine/Y
# Asparagine/N
# Glutamine/Q
# Aspartic Acid/D
# Glutamic Acid/E
#
# You can learn more about package authoring with RStudio at:
#
#   http://r-pkgs.had.co.nz/
#
# Some useful keyboard shortcuts for package authoring:
#
#   Build and Reload Package:  'Ctrl + Shift + B'
#   Check Package:             'Ctrl + Shift + E'
#   Test Package:              'Ctrl + Shift + T'

RemoveUnrecognizeAminoAcid <- function(seq_str) {
  #remove B, J, X, Z in seq
  seq_str =  gsub("B", "", seq_str)
  seq_str =  gsub("J", "", seq_str)
  seq_str =  gsub("X", "", seq_str)
  seq_str =  gsub("Z", "", seq_str)
  seq_str =  gsub("U", "", seq_str)
  seq_str =  gsub("O", "", seq_str)

  return(seq_str)
}
```

Cara Penggunaan ProtSegR

Package ProtSegR memiliki 3 fungsi utama yaitu:

1. ConvertToFeatureRepresentation
2. GenerateSegments
3. RemoveUnrecognizeAminoAcid

Untuk menggunakan ProtSegR, ikuti langkah-langkah berikut ini. Pertama adalah memuat package dengan perintah berikut.

```
library(ProtSegR)
```

RemoveUnrecognizeAminoAcid

Fungsi ini bertujuan untuk menghilangkan karakter yang bukan kode amino acid pada protein sequence. Berikut adalah contoh untuk menggunakan fungsi ini.

```
> clean_sequence = RemoveUnrecognizeAminoAcid("RSVTTEINTLFQTLTSIAEKVDPX")
> clean_sequence
[1] "RSVTTEINTLFQTLTSIAEKVDP"
```

GenerateSegments

Fungsi ini berguna untuk melakukan segmentasi sequence menjadi k adjacent segment dan k-1 overlapped segment. Untuk menggunakan fungsi ini dapat dilihat pada contoh berikut.

```
> GenerateSegments("RSVTTEINTLFQTLTSIAEKVDP", k=3)
             amino_acids
adjacent_1   "RSVTTEIN"
adjacent_2   "TLFQTLTS"
adjacent_3   "IAEKVDP"
overlapped_1 "TTEINTLFQ"
overlapped_2 "QTLTSIAEK"
```

ConvertToFeatureRepresentation

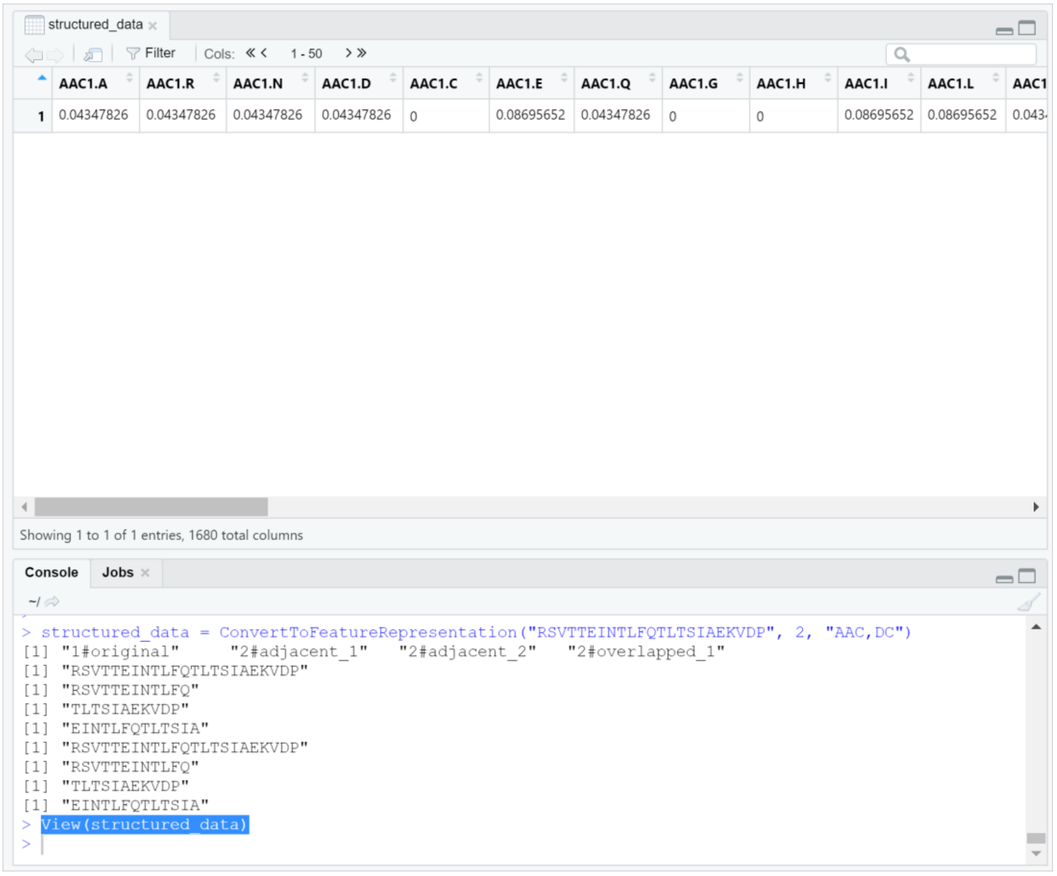
Fungsi ini dapat digunakan untuk proses ekstraksi fitur protein sequence dengan metode segmentasi dan menggunakan satu atau lebih protein descriptor. Untuk menggunakan fungsi ini dapat dilihat pada contoh di bawah ini.

```
> structured_data = ConvertToFeatureRepresentation("RSVTTEINTLFQTLTSIAEKVDP"
, 2, "AAC,DC")
[1] "1#original"     "2#adjacent_1"    "2#adjacent_2"    "2#overlapped_1"
[1] "RSVTTEINTLFQTLTSIAEKVDP"
[1] "RSVTTEINTLFQ"
[1] "TLTSIAEKVDP"
[1] "EINTLFQTLTSIA"
[1] "RSVTTEINTLFQTLTSIAEKVDP"
[1] "RSVTTEINTLFQ"
[1] "TLTSIAEKVDP"
[1] "EINTLFQTLTSIA"
```

Data terstruktur yang dibentuk oleh fungsi tersebut dapat dilihat dengan perintah berikut.

```
View(structured_data)
```

Hasilnya dapat dilihat seperti pada Gambar 2.



	AAC1.A	AAC1.R	AAC1.N	AAC1.D	AAC1.C	AAC1.E	AAC1.Q	AAC1.G	AAC1.H	AAC1.I	AAC1.L	AAC1
1	0.04347826	0.04347826	0.04347826	0.04347826	0	0.08695652	0.04347826	0	0	0.08695652	0.08695652	0.043

Showing 1 to 1 of 1 entries, 1680 total columns

```
> structured_data = ConvertToFeatureRepresentation("RSVTTEINTLFQTLTSIAEKVDP", 2, "AAC,DC")
[1] "1#original"      "2#adjacent_1"    "2#adjacent_2"    "2#overlapped_1"
[1] "RSVTTEINTLFQTLTSIAEKVDP"
[1] "RSVTTEINTLFQ"
[1] "TLTSIAEKVDP"
[1] "EINTLFQTLTSIA"
[1] "RSVTTEINTLFQTLTSIAEKVDP"
[1] "RSVTTEINTLFQ"
[1] "TLTSIAEKVDP"
[1] "EINTLFQTLTSIA"
> View(structured_data)
>
```

Gambar 2. Data terstruktur.