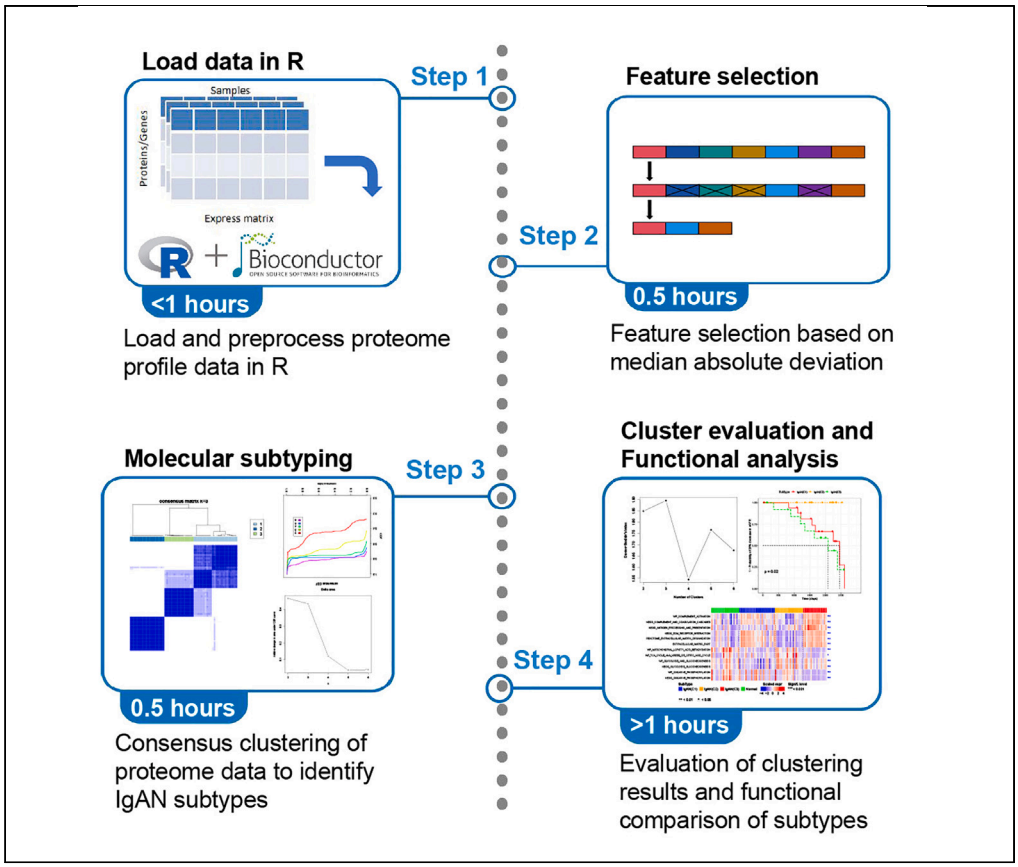
STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol for identifying and comparing molecular prognosis subtypes of IgAN using R



By providing a comprehensive view of protein dynamics, quantitative proteomics has emerged as a powerful tool for a better understanding of disease mechanisms. Here, we present a general workflow for identifying and comparing molecular subtypes of disease using proteomics data using R software. We describe steps for data preprocessing, feature selection, determination of subtypes, and functional interpretation of subtypes. These analyses can help us understand the nature of heterogeneous diseases, which is crucial for accurate diagnosis and personalized treatment.

Publisher's note: Undertaking any experimental protocol requires adherence to local institutional guidelines for laboratory safety and ethics.

Juan Wang, Yi Liu, Xizhao Chen, Mansheng Li, Yunping Zhu

limansheng@gmail.com (M.L.)
zhuyunping@gmail.com (Y.Z.)

Highlights

Steps of quality control and feature selection on proteomics data for IgAN subtyping

Steps for performing unsupervised consensus clustering analysis on IgAN samples

Steps for evaluating clustering results and determining IgAN subtypes

Steps for functional annotating and interpreting IgAN subtypes

Wang et al., STAR Protocols 5, 103138
September 20, 2024
https://doi.org/10.1016/j.xpro.2024.103138

STAR Protocols

CellPress
OPEN ACCESS

Protocol

Protocol for identifying and comparing molecular prognosis subtypes of IgAN using R

Juan Wang,[1,2] Yi Liu,[2] Xizhao Chen,[3] Mansheng Li,[2,4,*] and Yunping Zhu[1,2,5,*]

[1]Anhui Medical University, Hefei, Anhui 230000, China

[2]State Key Laboratory of Medical Proteomics, Beijing Proteome Research Center, National Center for Protein Sciences(Beijing), Beijing Institute of Lifeomics, Beijing 102206, China

[3]Department of Nephrology, First Medical Center of Chinese PLA General Hospital, Nephrology Institute of the Chinese People's Liberation Army, State Key Laboratory of Kidney Diseases, National Clinical Research Center for Kidney Diseases, Beijing Key Laboratory of Kidney Disease Research, Beijing 100853, China

[4]Technical contact

[5]Lead contact

*Correspondence: limansheng@gmail.com (M.L.), zhuyunping@gmail.com (Y.Z.)
https://doi.org/10.1016/j.xpro.2024.103138

SUMMARY

By providing a comprehensive view of protein dynamics, quantitative proteomics has emerged as a powerful tool for a better understanding of disease mechanisms. Here, we present a general workflow for identifying and comparing molecular subtypes of disease using proteomics data using R software. We describe steps for data preprocessing, feature selection, determination of subtypes, and functional interpretation of subtypes. These analyses can help us understand the nature of heterogeneous diseases, which is crucial for accurate diagnosis and personalized treatment.

For complete details on the use and execution of this protocol, please refer to Chen et al.[1]

BEFORE YOU BEGIN

Rapidly advancing precision medicine requires a systematic molecular-level interpretation of complex diseases. The development of high-throughput proteomics technology has made it possible to quantitatively and qualitatively analyze more than 10,000 proteins in biological samples.[2] The following protocol comprehensively analyzes IgAN samples and normal control samples based on quantitative proteomics methods, ultimately identify 3 prognosis-related subtypes. In addition to being applied to IgAN data, this protocol is also applicable to quantitative proteomic data from any sample or species to discover molecular subgroups. Before starting the protocol, please make sure to set up the computing environment by installing R, RStudio, BiocManager, ComplexHeatmap, ggplot2, clusterSim, fpc, survival, ConsensusClusterPlus, singscore and so on (key resources table).

Institutional permissions

The datasets used in this protocol to present the key steps are derived from our previous study.[1] These datasets are approved by the Research Ethics Committee (S2015-061-01). Users who adopt this protocol, please obtain permission from the relevant institutions before using these datasets.

Preparation: Software installation

Timing: <1 h

All analysis steps in this protocol are based on RStudio (2023.12.0 Build369) and implemented by R language (v4.3.2).

1. Download and install R, if you have not yet installed: https://cran.r-project.org/.
2. Download and install RStudio, if you have not yet installed: https://posit.co/products/open-source/rstudio/.
3. Download and install required R packages from Bioconductor (https://www.bioconductor.org/install/) and CRAN (https://cran.r-project.org/web/packages/).

```
>if (!require("BiocManager", quietly=TRUE))

install.packages("BiocManager")

>required.packages1 <- c("ComplexHeatmap", "ConsensusClusterPlus",

  "M3C", "singscore", "fgsea")

>BiocManager::install(required.packages1)

>required.packages2 <- c("ggplot2", "ggrepel", "ggcorrplot", "fpc",

  "car", "RColorBrewer", "circlize", "survival",

  "survminer", "FSA", "dunn.test", "clusterSim")

>install.packages(required.packages2)
```

Note: At present, there are three most commonly used R package installation methods: CRAN, Bioconductor and GitHub. When users install R packages, the system usually installs the latest version by default. However, previous versions can also be found on the official website, and this protocol specifies the specific R package version required in the key resources table.

⚠ CRITICAL: Errors often occur when installing packages, usually due to mismatched versions of Bioconductor and R.

4. Check whether all required R packages have been installed.

```
>required.packages <- c(required.packages1, required.packages2)

>installed <- sapply(required.packages, function(p) p %in% installed.packages())

>missing_packages <- required.packages[!installed]

>if (length(missing_packages) == 0) {

  print("All packages are installed")

} else {

  print(paste("The following packages are not installed:", paste(missing_packages,
collapse = ", ")))

}
```

Note: Each time the RStudio is restarted, the required package must be reloaded.

KEY RESOURCES TABLE

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Software and algorithms		
R (v4.3.2)	R Core Team[3]	https://cran.r-project.org
RStudio (2023.12.0 Build369)	RStudio Team[4]	https://posit.co/products/open-source/rstudio/

(*Continued on next page*)

Continued

REAGENT or RESOURCE	SOURCE	IDENTIFIER
BiocManager (v3.18)	Bioconductor (3.18)	https://www.bioconductor.org/install/
ComplexHeatmap (v2.18.0)	Bioconductor (3.18)	https://bioconductor.org/packages/release/bioc/html/ComplexHeatmap.html
ConsensusClusterPlus (v1.66.0)	Bioconductor (3.18)	https://bioconductor.org/packages/release/bioc/html/ConsensusClusterPlus.html
M3C (v1.24.0)	Bioconductor (3.18)	https://bioconductor.org/packages/release/bioc/html/M3C.html
singscore (v1.22.0)	Bioconductor (3.18)	https://bioconductor.org/packages/release/bioc/html/singscore.html
fgsea (v1.28.0)	Bioconductor (3.18)	https://bioconductor.org/packages/release/bioc/html/fgsea.html
ggplot2 (v3.4.4)	CRAN	https://CRAN.R-project.org/package=ggplot2
ggrepel (v0.9.4)	CRAN	https://CRAN.R-project.org/package=ggrepel
ggcorrplot(v0.1.4.1)	CRAN	https://CRAN.R-project.org/package=ggcorrplot
survminer (v0.4.9)	CRAN	https://CRAN.R-project.org/package=survminer
dunn.test (v1.3.5)	CRAN	https://CRAN.R-project.org/package=dunn.test
FSA (v0.9.5)	CRAN	https://CRAN.R-project.org/package=FSA
survival (v3.5-8)	CRAN	https://CRAN.R-project.org/package=survival
car (v3.1-2)	CRAN	https://CRAN.R-project.org/package=car
RColorBrewer (v1.1-3)	CRAN	https://CRAN.R-project.org/package=RColorBrewer
circlize (v0.4.16)	CRAN	https://CRAN.R-project.org/package=circlize
clusterSim (v0.51-3)	CRAN	https://CRAN.R-project.org/package=clusterSim
fpc (v2.2-11)	CRAN	https://CRAN.R-project.org/package=fpc
Deposited data		
Mass spectrometry data	Chen et al.[1]	iProX: PXD032710, https://www.iprox.cn

STEP-BY-STEP METHOD DETAILS

This protocol consists of four main analysis steps, which are divided into 6 sub-steps. We provide step-by-step details and timing for each sub-step, users can use RStudio (key resources table) to perform the following analysis process.

Loading and preprocessing of proteomic data

Timing: <30 min

The first step in data analysis and visualization in R is to load the data. With different data sources, users need to convert the raw dataset into a format suitable for downstream analysis. In this sub-step, we use the TMT quantitative proteomic data from previous study.[1] All 78 samples in the cleaned expression matrix were obtained from unique donors, with 19 tumor-adjacent-normal (5 cm far away from kidney tumor) and 59 IgAN biopsies (Lee's classification: III-IV).

1. Download the IgAN.zip from Zenodo (https://zenodo.org/records/10512182).

   Note: We have provided a zip file 'IgAN' (https://zenodo.org/records/10512182) on Zenodo (https://zenodo.org/), which contains five files and complete code for this protocol. The file "mergeSampleRatio_90.txt" is the table of protein quantitative data, the file "uniprot-your-list.csv" is the protein functional annotation information, the file "eGFR clinical follow-up raw data.csv" is the clinical follow-up raw data of estimated glomerular filtration rate, the file "clinical data for IgAN.csv" is the important clinical parameters of different IgAN patients, and the file "IgAN pathways.gmt" is the gene list set for singscore in functional analysis.

2. Create a new R script file using "New File" menu in RStudio.
3. Set the working directory to the same path where the data files are stored.

```
>dir_root <- "D:/Workplace/IgAN"

>setwd(dir_root)
```

Note: Users need to set the working directory to the same path where the data files are stored.

4. Create a subdirectory named "result_QC/" in the current working directory.

```
>dir_QC <- "result_QC/"

>dir.create(file.path(dir_QC), showWarnings = FALSE)
```

5. Load raw data file of proteome.

```
>eRaw <- read.table("data/mergeSampleRatio_90.txt", header=T, row.names=1, stringsAsFac-
tors=F, check.names=F)

>acc2sym <- read.csv("data/uniprot-yourlist.csv", header=T, row.names=2, check.names=F)
```

Note: The "mergeSampleRatio_90.txt" file contains the protein abundance of 90 samples, in which, the first row is sample names and the first column is the accession number of proteins from UniProt database. The "uniprot-yourlist.csv" file contains the annotation information of the proteins detected from the 90 samples.

6. Replace the UniProt accession number of the protein with the corresponding Gene Symbol.

```
>acc2sym <- acc2sym["Gene names (primary)"]

>colnames(acc2sym) <- c("Symbol")

>acc2sym$Symbol <- sub(";.+", "", acc2sym$Symbol)

>eSet <- eRaw

>eSet$Symbol <- acc2sym[rownames(eSet), ]

>eSet[is.na(eSet$Symbol)|eSet$Symbol=="", ]$Symbol <- rownames(eSet[is.na(eSet$Symbol)|
eSet$Symbol=="", ])

>rownames(eSet) <- make.unique(as.character(eSet$Symbol))

>eSet <- eSet[ , -ncol(eSet)]
```

Note: The UniProt database (https://www.uniprot.org/) can perform ID mapping on the UniProt ID of each protein to obtain corresponding annotation information, including Entry name, Protein name, Gene name, Organism, and Length.

Note: Considering the efficiency and readability of subsequent analysis, users need to convert the ID in the raw data. In this protocol, we choose to convert UniProt ID to Gene Symbol, and when multiple UniProt ID correspond to one Gene Symbol, we can use the make.unique() function to add a serial number to Gene Symbol to make it unique.

7. Group samples by type.

```
>norm_p <- grep("Nor", colnames(eSet), value=T)
>iga_p <- grep("IgA", colnames(eSet), value=T)
>mn_p <- grep("MN", colnames(eSet), value=T)
>sample_p <- c(norm_p, iga_p, mn_p)
>Group_p <- c("Normal", "IgAN", "MN")
>type_p <- c(rep("Normal", length(norm_p)), rep("IgAN", length(iga_p)), rep("MN", length(mn_p)))
>cols_p <- setNames(c("green", "red3", "orange"), Group_p)
>sampleType_p <- data.frame(sample_p, type_p, check.names=F)
>rownames(sampleType_p) <- sample_p
```

Note: The grep() function can match the corresponding element vectors according to a given string. In this protocol, we divided 90 samples into 3 groups (normal, IgA nephropathy, membranous nephropathy) according to the sample type.

8. Filter samples based on missing values and membranous nephropathy.

```
>eSet <- eSet[apply(eSet, 1, function(x) !all(is.na(x))), ]
>delete.na5 <- function(DF) {
df1 <- DF[ ,grep('IgA', names(DF), value=TRUE)]
df2 <- DF[ ,grep('MN', names(DF), value=TRUE)]
df3 <- DF[ ,grep('Nor', names(DF), value=TRUE)]
DF[rowSums(is.na(df1)) <= 54 & rowSums(is.na(df2)) <= 7 & rowSums(is.na(df3)) <= 14, ]
}
>eSet.F <- delete.na5(eSet)
>drop <- mn_p
>eSet.F <- eSet.F[ ,-which(colnames(eSet.F) %in% drop)]
```

Note: In the sample, not all data are valid, users must filter them. In this protocol, the first is row processing: delete rows with all missing values in the eSet data frame, and filter specific rows by limiting the total number of missing values in rows through the constructed delete.na5() function. The second is column processing: remove samples of membranous nephropathy.

9. Log-transform and normalize the proteome profile data.

```
>eSet.F[eSet.F==100] <- NA
>eSet.F[eSet.F==0.01]<- NA
>eSet2 <- log2(eSet.F)
>eSet2.N <- data.frame(scale(eSet2), check.names = F)
```

Note: During logarithmic conversion, in order to prevent errors caused by taking the logarithm of zero, users can replace zero with NA or add one to all constants.

10. Impute the missing values with the minimum values in corresponding columns.

```
>imput_min <- function(df){
  apply(df, 2, function(x){
    ifelse(is.na(x), min(x, na.rm = TRUE), x)
  })
  df <- data.frame(df, check.names=F)
  return(df)
}
>eSet2.NI <- imput_min(eSet2.N)
>eSet3 <- eSet2.NI
```

Quality control analysis of proteomic data

Timing: <30 min

While screening and normalizing the data, quality control analysis (QC) is also essential. This QC step directly determines whether the subsequent analysis is reliable. Currently, there are various visualization methods for quality control analysis of proteomics data. In this sub-step, we select boxplot, line plot, and correlation plot to determine the reliability and accuracy of the data. Users can choose these QC methods we mentioned according to their needs.

11. Boxplot visualization of log2 transformed proteome data.

```
>data_bx <- eSet
>data_bx = log(data_bx, 2)
>boxplot(data_bx, col="blue", ylab="log2(S/N Ratio)", las = 2, cex.axis=0.8)
>dev.print(png, paste0(dir_QC, "boxplot.png"), width=12, height=5, units="in", res=300)
>dev.print(pdf, paste0(dir_QC, "boxplot.pdf"), width=12, height=5)
>dev.off()
```

Note: The boxplot can show the distribution of data and identify outliers (Figure 1A). The methods for handling outliers include: direct deletion, processing as missing values, mean value correction, no processing, etc. As shown in Figure 1A, there are outliers beyond the upper and lower boundaries. However, for the accuracy of later typing, we did not process outliers because these outliers may be feature proteins of each sample for subsequent typing.

Note: To prevent subsequent visualization errors, users can turn off the graphics device using the dev.off() function, and the next opened device will become the current device.

12. Count the accumulated number of proteins identified in all collected samples and display them in a line plot.

```
>library(ggplot2)

>library(ggrepel)

>colCnt <- apply(eSet, 2, function(x){sum(!is.na(x))})

>colCnt <- colCnt[order(colCnt, decreasing=F)]

>df_cnts <- colCnt[1]

>for (i in 2:length(colCnt)) {

  print(i)

  tmp_df <- data.frame(eSet[ ,names(colCnt)[1:i]])

  tmp_df <- tmp_df[apply(tmp_df, 1, function(x) !all(is.na(x))), ]

  df_cnts[i] <- nrow(tmp_df)

}

>df_cnts <- data.frame(Ind=c(1:length(df_cnts)), Count=df_cnts)

>df_cnts1 <- df_cnts[which(df_cnts$Ind==1), ]

>df_cnts2 <- df_cnts[which(df_cnts$Ind==round(ncol(eSet)*0.25)), ]

>df_cnts3 <- df_cnts[which(df_cnts$Ind==round(ncol(eSet)*0.5)), ]

>df_cnts4 <- df_cnts[which(df_cnts$Ind==round(ncol(eSet)*0.75)), ]

>df_cnts5 <- df_cnts[which(df_cnts$Ind==ncol(eSet)), ]

>ggplot(df_cnts, aes(x=Ind, y=Count)) +

geom_bar(stat="identity", width=.1) +

geom_point() +

geom_line(group = 1, colour ="blue", size =1) +

  scale_x_discrete(expand = c(0,0), breaks=seq(0, nrow(df_cnts), by=5), limits=c(1:nrow(df_cnts)), labels=seq(0, nrow(df_cnts), by=5)) + xlab("Number of samples") +
ylab("Number of identified proteins") +

geom_text_repel(data=df_cnts1, aes(x=Ind, y=Count, label=Count), nudge_x=.5, nudge_y=50,
min.segment.length = unit(0, 'lines')) +

  geom_text_repel(data=df_cnts2, aes(x=Ind, y=Count, label=paste0(Count, ", ", Ind, "
(25%)")), nudge_x=.5, nudge_y=50, min.segment.length = unit(0, 'lines')) +

  geom_text_repel(data=df_cnts3, aes(x=Ind, y=Count, label=paste0(Count, ", ", Ind, "
(50%)")), nudge_x=.5, nudge_y=50, min.segment.length = unit(0, 'lines')) +

  geom_text_repel(data=df_cnts4, aes(x=Ind, y=Count, label=paste0(Count, ", ", Ind, "
(75%)")), nudge_x=.5, nudge_y=50, min.segment.length = unit(0, 'lines')) +

  geom_text_repel(data=df_cnts5, aes(x=Ind, y=Count, label=Count), nudge_x=.5, nudge_y=50, min.segment.length = unit(0, 'lines')) +

  theme_bw() +

  theme(axis.text.x = element_text(size=6, angle=90, vjust =0.5, hjust=1), axis.text.y =
element_text(size=9))

>ggsave(paste0(dir_QC, "ProteinAccumulation.png"), width=5, height=4)

>ggsave(paste0(dir_QC, "ProteinAccumulation.pdf"), width=5, height=4)
```

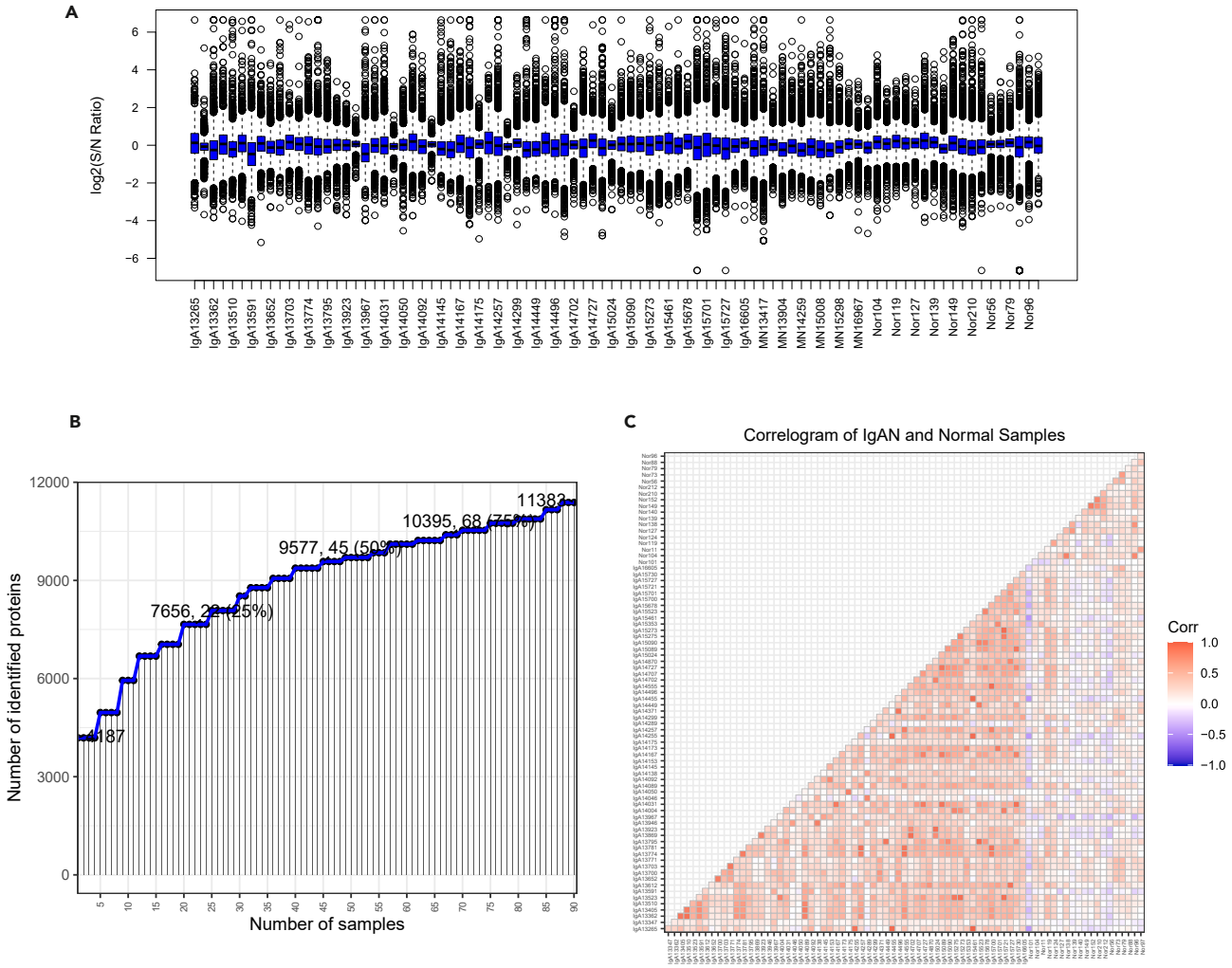


Figure 1. The quality control of MS data and downstream bioinformatics analysis

(A) Explore the distribution and outliers of mass spectrometry data with boxplots.

(B) Number of proteins identified for different sample numbers. As the number of samples changes, the number of identified proteins increases, and finally saturation occurs.

(C) Correlation coefficient plot among IgAN and normal control samples. There is a strong positive correlation between IgAN samples. But the correlation between IgAN and normal samples is not very large, or even negative.

Note: The line plot can show the trend of the accumulated number of identified proteins in the samples (Figure 1B). From Figure 1B, we can observe that as the number of samples changes, the number of identified proteins increases, and finally saturation occurs.

13. Calculate the correlation coefficient between samples and visualize the results.

```
>library(ggcorrplot)

>data_cor <- eSet3

>temp_cor <- cor(data_cor, use="pairwise.complete.obs")

>ggcorrplot(temp_cor, hc.order = F,

  type = "lower",
```

```
  lab_size = 3,
  method="square",
  colors = c("blue", "white", "tomato2"),
  title="Correlogram of IgAN and Normal Samples",
  tl.srt = 90,
  tl.cex = 4,
  ggtheme=theme_bw) +
  theme(plot.title = element_text(size = 12, hjust = 0.5))
>ggsave(paste0(dir_QC, "Corrplot.png"), width = 6, height = 6)
>ggsave(paste0(dir_QC, "Corrplot.pdf"), width = 6, height = 6)
```

Note: The cor() function can calculate the correlation coefficient between all variables in pairs. Usually the Pearson correlation coefficient is selected by default; users can choose the type of correlation coefficient as needed.

Note: In this protocol, we first use the cor() function to obtain the correlation matrix between IgAN and normal samples, and then use the ggcorrplot() function to intuitively present the degree of correlation among different samples (Figure 1C). It can be seen from the figure that there is a strong positive correlation between IgAN samples. But the correlation between IgAN and normal samples is not very large, or even negative.

Feature selection of proteomic data

Timing: <30 min

In order to improve the computational efficiency and accuracy of the subtyping algorithms, it is essential to filter the obviously discrete molecular features of different samples. In this sub-step, we use median absolute difference (MAD) to sort features, and select the top 800 important protein features after comprehensively considering the sample size and feature scale.

14. Filter and normalize the proteomic data that used to cluster.

```
>data_iga <- eSet2[,iga_p]
>data_iga[is.na(data_iga)] <- 0
>df_mad <- apply(data_iga, 1, mad)
>df_mad800 <- data_iga[rev(order(df_mad))[1:800], ]
>df_median <- sweep(df_mad800, 1, apply(df_mad800, 1, median, na.rm=T))
```

Note: When selecting proteome data features, the features usually refer to various measurements and indicators used to describe the expression level or properties of proteins, such as protein quantitative data, mass spectrometry data and structural characteristics. In this protocol, the features are quantitative data of proteins.

Note: In addition to using MAD, the coefficient of variation (CV) and standard deviation (SD) can also be calculated for feature selection. Users can flexibly select the percentage of features based on the sample size and feature scale.

Consensus cluster analysis of proteomic data

Timing: <30 min

Identifying disease subtypes is an essential part of exploring the molecular heterogeneity of a disease. In this sub-step, we performed unsupervised consensus clustering analysis on all IgAN samples based on their proteomic data, dividing 59 IgAN samples into different subtypes.

15. Create a subdirectory named "result_subtype/" in the current working directory.

```
>dir_subtype <- "result_subtype/"

>dir.create(file.path(dir_subtype), showWarnings = FALSE)
```

16. Consensus clustering analysis of proteome data using the ConsensusClusterPlus Package.

```
>library(ConsensusClusterPlus)

>clusters <- ConsensusClusterPlus(as.matrix(df_median), maxK=6, reps=1000, pItem=0.8,
pFeature=0.8, distance="pearson", clusterAlg="hc", title='result_subtype', plot="png")

>subtypes_list <- list()

>for(k in 2:6) {

subtypes <- data.frame(Class=clusters[[k]]$consensusClass [(order(clusters[[k]]
$consensusClass))])

rownames(subtypes) <- names(clusters[[k]]$consensusClass [(order(clusters[[k]]$consensus
Class))])

  subtypes$Class=paste('IgAN(C', subtypes$Class, ")", sep='')

  table(subtypes)

  subtypes_list[[k]] <- subtypes

  write.csv(subtypes, paste0(dir_subtype,"consensusClass_",k,".csv"))

}
```

Note: Currently there is a ready-made ConsensusClusterPlus package for consensus clustering. Users can modify the numerical or character values of different parameters in the ConsensusClusterPlus() function to obtain the classification status under different subtype (K value) results.

Note: In this protocol, the clustering method we choose is hierarchical clustering (hc), the sampling ratio is 0.8, and the number of resampling is 1000. The output results are classifications with K ranging from 2 to 6, and saved in the subdirectory folder in the form of PNG images (Figure 2). According to Figure 2, the matrix heatmap is relatively clean when k is 4 or 5, and the relative change value of the area under the CDF curve does not increase significantly when k is 4. These results provide a reference for determining the K value in the next sub-step.

Determination of the number of disease molecular subtypes

Timing: <1 h

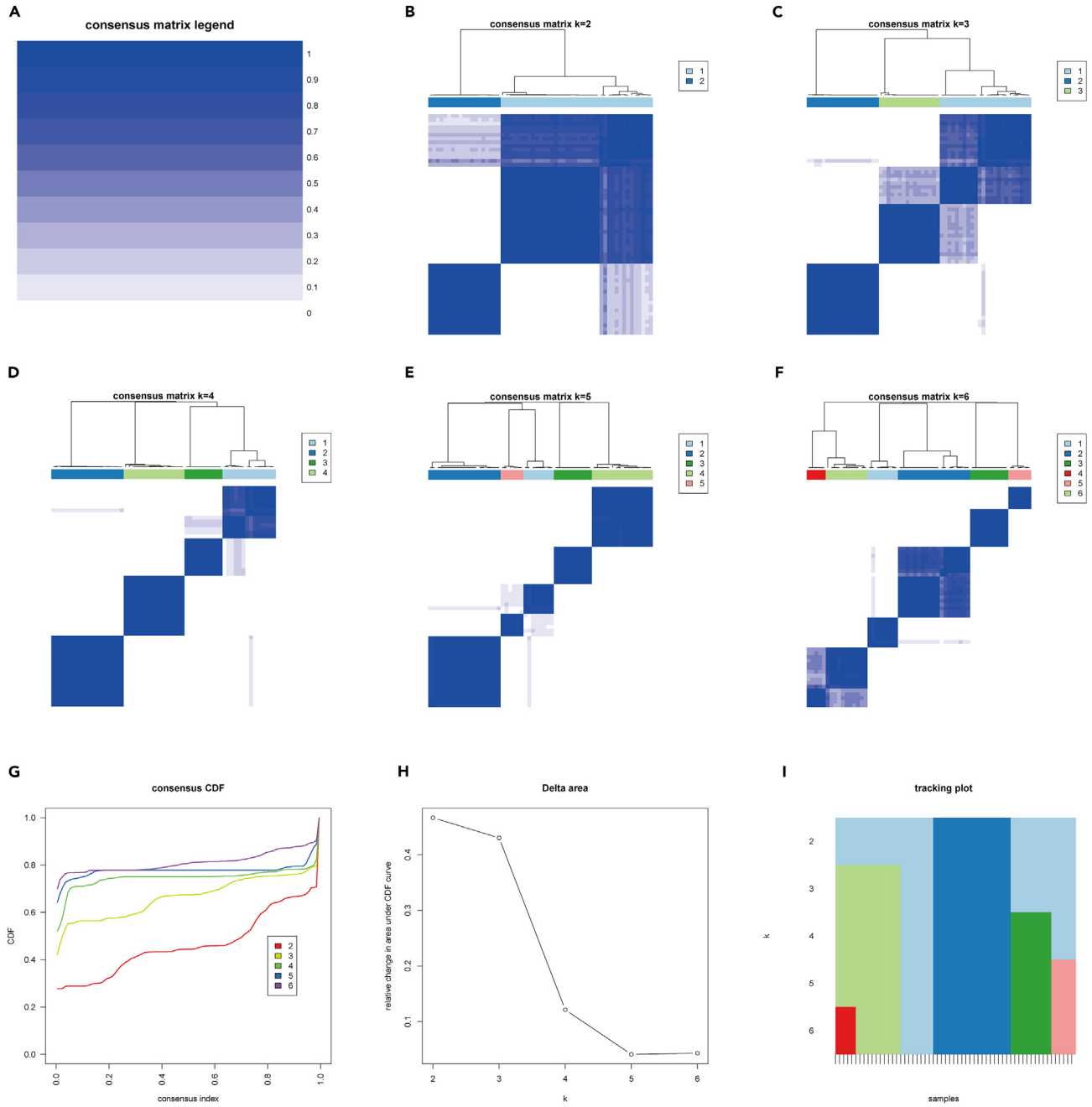


Figure 2. The consensus clustering analysis of 59 IgAN samples

(A) Consensus matrix legend. The darker the color, the higher the purity of typing.

(B–F) Consensus clustering matrix heatmap when k is 2∼6.

(G) Cumulative Distribution Function (CDF) curve showing the sampling error in different classifications.

(H) Delta Area Plot for observing the relative change of the area under the CDF curve.

(I) Tracking plot for k from 2 to 6, reflecting the stability between samples after different classification.

The number of molecular subtypes can be determined according to the results of the consensus clustering analysis: the K value is the best when the heatmap is clean, the CDF decline slope is small, and the relative change value of the area under the CDF curve does not increase significantly.[5] In addition, the optimal K value can also be determined by automated methods (e.g., PAC method,[6] GAP

method,[7] M3C method,[8] etc.), internal evaluation indicators (e.g., Silhouette Coefficient, Calinskin-Harabasz Index, Davies-Bonldin Index, etc.), and clinical prognostic analysis. In this sub-step, we use the M3C package, internal evaluation indicators and survival analysis to determine the optimal K value, and finally classify the 59 IgAN samples into three subtypes.

17. Determine the K value through the M3C package.

```
>library(M3C)
>res <- M3C(as.matrix(df_median), maxK = 6,
  pItem=0.8, repsreal = 1000, clusteralg = "hc",
  pacx1 = 0.1, pacx2 = 0.9, objective = "PAC")
```

Note: M3C is a Monte Carlo reference-based consensus clustering algorithm that can select K values by relative clustering stability index (RCSI) and proportion of ambiguous clustering (PAC) score.[8] In this protocol, the best K value we obtained based on the M3C function is 5.

Note: In the M3C() function, Most of the parameter values we used are the same as in sub-step 4b. Users can modify the parameter values by themselves.

18. Evaluate the K value through internal evaluation indicators.

```
>for(k in 2:6) {
assign(paste0("class", k), data.frame(Class=clusters[[k]]$consensusClass))
}
>library(clusterSim)
>DB <- numeric(5)
>for(k in 2:6) {
  class <- get(paste0("class", k))
  DB[k-1]      <-      index.DB(t(df_median),      class$Class,      dist(t(df_median)),
centrotypes="medoids")$DB
}
>plot(x = 2:6, y = DB, type = "b", pch = 19, xlab = "Number of Clusters", ylab = "Davies-Bouldin's
Index")
>dev.print(png, paste0(dir_subtype, "DB.png"), width = 6, height = 6, units = "in", res = 300)
>dev.print(pdf, paste0(dir_subtype, "DB.pdf"), width = 6, height = 6)
>library(fpc)
>CH <- numeric(5)
>for(k in 2:6) {
  class <- get(paste0("class", k))
  CH[k-1] <- calinhara(t(df_median), class$Class)
}
```

```
>plot(x = 2:6, y = CH, type = "b", pch = 19, xlab = "Number of Clusters", ylab = "Calinski-Harabasz Index")

>dev.print(png, paste0(dir_subtype, "CH.png"), width = 6, height = 6, units = "in", res = 300)

>dev.print(pdf, paste0(dir_subtype, "CH.pdf"), width = 6, height = 6)
```

Note: The evaluation index of clustering results usually can be divided into internal indicators and external indicators. The internal indicators can evaluate the clustering results by the compactness and separation degree of the samples without the help of real labels. In this protocol, we choose the K value when the Calinskin-Harabasz Index is maximum or Davies-Bonldin Index is minimum. As shown in Figure 3, we can observe that the clustering works best when K = 4.

19. Determine the final K value again based on survival analysis.
    a. Reading and processing of clinical follow-up data.

```
>clinic <- read.csv('data/eGFR clinical follow-up raw data.csv', header=T, row.names=2, check.names=F)

>sampleInfo <- read.csv("data/clinical data for IgAN.csv", header = T, row.names=5, check.names = F, fileEncoding = "latin1")

>sampleInfo <- sampleInfo[grep("IgA", sampleInfo$Pathological.number2), ]

>sampleInfo <- subset(sampleInfo, select = Pathological.number2)

>clinic$sample <- sampleInfo[rownames(clinic), ]

>clinic.part <- data.frame(sample=clinic$sample, Status=clinic$Status, Days=clinic$Days)

>clinic.part <- clinic.part[!is.na(clinic.part$Status), ]

>clinic.part$Class3 <- subtypes_list[[3]][clinic.part$sample, ]

>clinic.part$Class4 <- subtypes_list[[4]][clinic.part$sample, ]

>clinic.part$Class5 <- subtypes_list[[5]][clinic.part$sample, ]

>clinic.part$Class6 <- subtypes_list[[6]][clinic.part$sample, ]
```

Note: Survival analysis is a common method to evaluate the prognostic value, which needs to combine the follow-up results and follow-up time of patients for analysis. The "eGFR clinical follow-up raw data.csv" file is the clinical follow-up raw data of estimated glomerular filtration rate (eGFR), including patients' serum creatinine (Scr) levels, eGFR values, survival status, survival time, etc. The "clinical data for IgAN.csv" file contains important clinical parameters of different IgAN patients, such as pathology number, pathological stage, gender, age, blood pressure, etc.

    b. Survival analysis and results visualization for disease subtypes.

```
>library(survival)

>library(survminer)

>fit3 <- survfit(Surv(Days, Status==1) ~ Class3, data = clinic.part)

>summary(fit3)

>ggsurvplot(fit3,
```

```
  pval = TRUE, conf.int = F,
  risk.table = F,
  cumevents = F,
  linetype = "strata",
  surv.median.line = "hv",
  ggtheme = theme_bw(),
  palette = c("red", "orange", "green"),
  xlab = "Time (days)",
  ylab = "1 - Probability of 30% decrease in eGFR",
  legend.title="Subtype",
  legend.labs =c("IgAN(C1)", "IgAN(C2)", "IgAN(C3)"),
  xlim=c(0,3000)
)
>dev.print(png, paste0(dir_subtype,"Prob_curve_eGFR3.png"), width=6, height=6, units="in", res=300)
>dev.print(pdf, paste0(dir_subtype,"Prob_curve_eGFR3.pdf"), width=6, height=6)
>fit4 <- survfit(Surv(Days, Status==1) ~ Class4, data = clinic.part)
>summary(fit4)
>ggsurvplot(fit4,
  pval = TRUE, conf.int = F,
  risk.table = F,
  cumevents = F,
  linetype = "strata",
  surv.median.line = "hv",
  ggtheme = theme_bw(),
  palette = c("red", "orange", "green", "blue"),
  xlab = "Time (days)",
  ylab = "1 - Probability of 30% decrease in eGFR",
  legend.title="Subtype",
  legend.labs =c("IgAN(C1)", "IgAN(C2)", "IgAN(C3)", "IgAN(C4)"),
  xlim=c(0,3000)
)
>dev.print(png, paste0(dir_subtype,"Prob_curve_eGFR4.png"), width=6, height=6, units="in", res=300)
>dev.print(pdf, paste0(dir_subtype,"Prob_curve_eGFR4.pdf"), width=6, height=6)
>fit5 <- survfit(Surv(Days, Status==1) ~ Class5, data = clinic.part)
>summary(fit5)
>ggsurvplot(fit5,
  pval = TRUE, conf.int = F,
```

```
  risk.table = F,
  cumevents = F,
  linetype = "strata",
  surv.median.line = "hv",
  ggtheme = theme_bw(),
  palette = c("red", "orange", "green", "blue", "purple"),
  xlab = "Time (days)",
  ylab = "1 - Probability of 30% decrease in eGFR",
  legend.title="Subtype",
  legend.labs =c("IgAN(C1)", "IgAN(C2)", "IgAN(C3)", "IgAN(C4)", "IgAN(C5)"),
  xlim=c(0,3000)
)
>dev.print(png, paste0(dir_subtype,"Prob_curve_eGFR5.png"), width=6, height=6, units="in", res=300)
>dev.print(pdf, paste0(dir_subtype,"Prob_curve_eGFR5.pdf"), width=6, height=6)
>fit6 <- survfit(Surv(Days, Status==1) ~ Class6, data = clinic.part)
>summary(fit6)
>ggsurvplot(fit6,
  pval = TRUE, conf.int = F,
  risk.table = F,
  cumevents = F,
  linetype = "strata",
  surv.median.line = "hv",
  ggtheme = theme_bw(),
  palette = c("red", "orange", "green", "blue", "purple", "brown"),
  xlab = "Time (days)",
  ylab = "1 - Probability of 30% decrease in eGFR",
  legend.title="Subtype",
  legend.labs =c("IgAN(C1)", "IgAN(C2)", "IgAN(C3)", "IgAN(C4)", "IgAN(C5)", "IgAN(C6)"),
  xlim=c(0,3000)
)
>dev.print(png, paste0(dir_subtype,"Prob_curve_eGFR6.png"), width=6, height=6, units="in", res=300)
>dev.print(pdf, paste0(dir_subtype,"Prob_curve_eGFR6.pdf"), width=6, height=6)
```

Note: For survival analysis, users first need to create a surv object using the Surv() function, and then calculate the survival rate using the survfit() function. Here, we fit the time and survival status to calculate survival rate by Kaplan-Meier estimation method. And visualize the survival curve through the ggsurvplot() function in the survminer package.

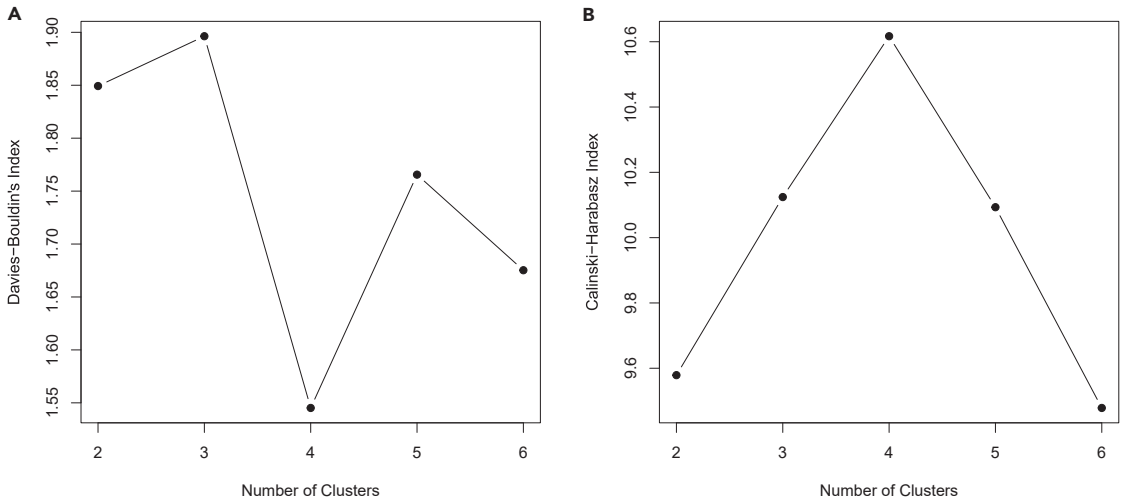


Figure 3. Evaluate different IgAN clustering results using internal indicators

(A) Numerical plot of Davies-Bonldin Index when k is 2 to 6. The K value is the best when the Davies-Bonldin Index is minimum.

(B) Numerical plot of Calinskin-Harabasz Index when k is 2 to 6. The K value is the best when the Calinskin-Harabasz Index is maximum.

Note: Currently, there is no standard procedure to definitively determine the optimal K value. Since the k values we obtained in step 17 and 18 are inconsistent, in order to more comprehensively evaluate which K value is the best, we further add the classification results of K = 3 and 6 for survival analysis(Figure 4). It can be seen from the figure that when k = 3, 4, 5 and 6, the survival differences between different subtypes are significant (P values are less than 0.05). But when k = 3, the P value is the smallest. Considering that molecular typing is for accurate diagnosis and personalized treatment, we choose to focus on clinical prognosis and divide IgAN samples into three subtypes (C1, C2, and C3).

Functional analysis of the disease subtypes

Timing: <30 min

Many studies have failed to further analyze the function of subtypes after identifying disease subtypes. In this sub-step, we scored the gene set based on proteomic data and implemented heatmap visualization after obtaining differential expression pathways between different subtypes using analysis of variance (ANOVA) or Kruskal-Wallis non-parametric test.

20. Create a subdirectory named "result_singscore/" in the current working directory.

```
>dir_singscore <- "result_singscore/"
>dir.create(file.path(dir_singscore), showWarnings = FALSE)
```

21. Reading and processing of disease typing information.

```
>Subtypes <- read.csv("result_subtype/consensusClass_3.csv", header=T, stringsAsFactors=F)
>rownames(Subtypes) <- Subtypes[ ,1]
>c1 <- rownames(Subtypes[Subtypes$Class=="IgAN(C1)", ])
>c2 <- rownames(Subtypes[Subtypes$Class=="IgAN(C2)", ])
```

```
>c3 <- rownames(Subtypes[Subtypes$Class=="IgAN(C3)", ])

>sample <- c(norm_p,c1,c2,c3)

>type <- c(rep("Normal", length(norm_p)), rep("IgAN(C1)", length(c1)), rep("IgAN(C2)",
length(c2)), rep("IgAN(C3)" , length(c3)))

>Group <- c("Normal", "IgAN(C1)", "IgAN(C2)", "IgAN(C3)")

>sampleType <- data.frame(sample=sample, type=type, check.names=F)

>rownames(sampleType) <- sample

>cols <- setNames(c("green", "blue", "orange", "red"), Group)
```

Note: The disease subtyping information here is the result of the previous consensus cluster analysis (step 16). We divided the 78 samples into 4 groups (Normal Control, IgAN-C1, IgAN-C2, IgAN-C3) according to the subtyping results.

22. Pathway score analysis of typing samples using the singscore package.
    a. Import the gene set and score the activity of pathways in the 4 different groups of samples.

```
>library(singscore)

>gSet <- fgsea::gmtPathways("data/IgAN pathways.gmt")

>data_ss <- eSet3[ ,sample]

>data_ss <- data_ss[apply(data_ss, 1, function(x) !all(is.na(x))),]

>data_ss[is.na(data_ss)] <- 0

>rankedData <- rankGenes(data_ss)

>scoredf <- data.frame()

>terms <- NULL; counts <- NULL; hits <- NULL; genes <- NULL

>for(i in 1:length(gSet)){

  print(paste0(i-1, "/", length(gSet)-1))

  score <- simpleScore(rankData = rankedData, upSet = gSet[[i]],

    centerScore = T, knownDirection = T)

  scores <- score$TotalScore

  if(length(scores)!=0){

    terms <- rbind(terms, sub("^.+?_MM_", "", names(gSet[i])))

    counts <- rbind(counts, length(gSet[[i]]))

    tmp_hit <- rownames(data_ss) %in% gSet[[i]]

    hits <- rbind(hits, sum(tmp_hit==T))

    genes <- rbind(genes, paste(rownames(data_ss)[which(tmp_hit)], collapse=", "))

    scoredf <- rbind(scoredf, scores)

  }

}

>rownames(scoredf) <- terms

>colnames(scoredf) <- colnames(data_ss)
```

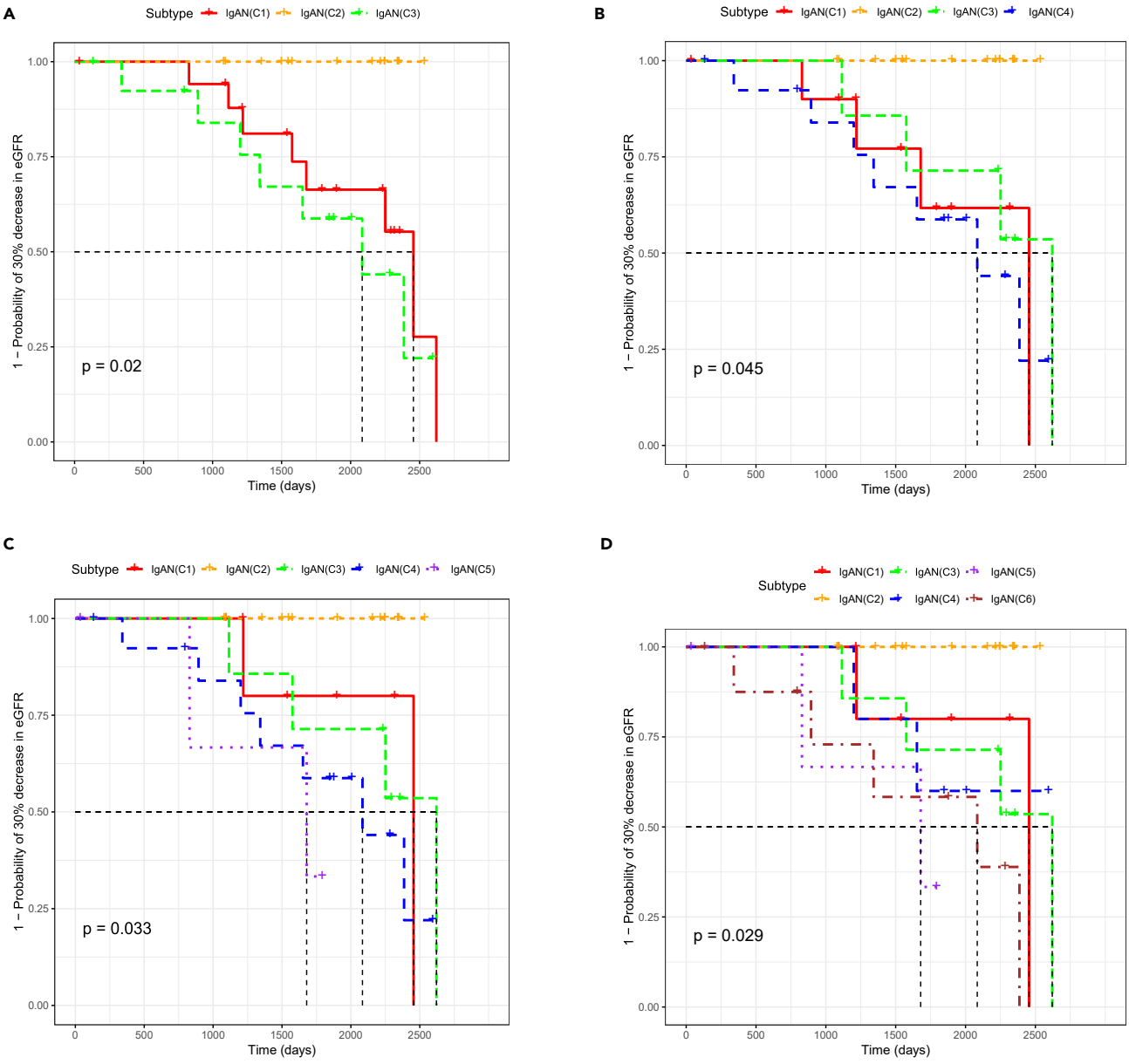


Figure 4. The prognostic value of different IgAN subtypes when k is 3 to 6

(A–D) Kaplan–Meier renal survival curves of IgAN subtypes according to 30% eGFR decline as the composite endpoint. If the P value is less than 0.05, it means that there is a significant difference in survival rate between different subtypes.

Note: Singscore is a single-sample scoring method based on ranking. It first utilizes the rankGenes() function to rank the gene expression intensities in each sample, and then uses the ranking to score the gene set pathway for each sample.

b. Analysis of variance or Kruskal-Wallis non-parametric test for enriched pathways.

```
>library(car)
>library(FSA)
>library(dunn.test)
>data <- scoredf
```

```
>P.Value <- NULL; t.type <- NULL
>for(i in 1:nrow(data)){
  print(i)
  tmp_data <- t(data[i,])
  tmp_data <- tmp_data[which(is.na(tmp_data) == 0),]
  group <- factor(sampleType[which(sample %in% names(tmp_data)),]$type, levels = Group)
  t <- shapiro.test(tmp_data)
  tmp_data <- reshape2::melt(tmp_data)
  tmp_data$group <- group
  f <- leveneTest(value~group, tmp_data)
  if(t$p.value>0.05){
    model <- aov(value ~ group, tmp_data)
    fit_a<-summary(model)
    P.Value[i] <- fit_a[[1]]["Pr(>F)"][1,1]
    t.type[i] <- 'ANOVA, type I SS'
  }else{
    fit_k <- kruskal.test(tmp_data$value, group)
    P.Value[i] <- fit_k$p.value
    t.type[i] <- 'Kruskal test'
  }
}
>res <- data.frame(P.Value, adj.P.Val=p.adjust(P.Value, method='BH'), t.type,
check.names=F)
>res2 <- data.frame(data, P.Value, adj.P.Val=p.adjust(P.Value, method='BH'), t.type,
check.names=F)
```

Note: To determine whether there are significant differences in the expression scores of enriched pathways among different subtypes, we choose analysis of variance or Kruskal-Wallis non-parametric test for P value calculation.

c. Heatmap visualization of the difference analysis results of singscore.

```
>data_m <- as.matrix(scoredf)
>data_scaled = t(scale(t(data_m)))
>library(ComplexHeatmap)
>library(RColorBrewer)
>ha <- HeatmapAnnotation(
  SubType = type,
  col = list(SubType = cols),
show_annotation_name = FALSE,
```

```
annotation_height = c(8,8), gp = gpar(lwd = .5, col = "white"),
  annotation_legend_param = list(nrow=1, title_position = "topleft")
)
>ht <- Heatmap(
  data_scaled,
  name="ht",
  width = unit(12, "cm"),
  cluster_rows = F,
  clustering_distance_rows = "pearson",
  clustering_method_rows = "complete",
  row_names_side = "left",
  row_names_gp = gpar(fontsize = 6.5),
  cluster_columns = F,
  show_column_names = F,
  column_names_side = "bottom",
  column_names_gp = gpar(fontsize = 6),
  column_title = NULL,
  na_col = "white",
  rect_gp = gpar(lwd = .01, col = "white"),
  top_annotation = ha,
  heatmap_legend_param=list(title="Scaled expr", title_position = "topcenter",legend_-
direction = "horizontal"
  )
)
>Signif2 <- symnum(res$P.Value, corr = FALSE, na = FALSE,
    cutpoints = c(0, 0.001, 0.01, 0.05, 0.1, 1),
    symbols = c(" ***", " **", " *", " .", ""))
>pvalue <- res$P.Value
>is_sig = pvalue < 0.01
>pch = rep("*", length(pvalue))
>pch[!is_sig] = NA
>library(circlize)
>pvalue_col_fun = colorRamp2(c(0,2,3), c("green","white","red"))
>ra = rowAnnotation(pvalue = anno_text(Signif2, gp = gpar(fontsize = 8, col="blue", border =
"white"), location = 0.0, just = "left"),
width = max_text_width(Signif2)*1.2, show_annotation_name = F)
>lgd_sig = Legend(title = "Signif. level", pch = "***", type = "points", labels = "< 0.001")
>lgd_sig2 = Legend(title = " ", pch = "**", type = "points", labels = "< 0.01")
```

```
>lgd_sig3 = Legend(title = " ", pch = "*", type = "points", labels = "< 0.05")

>draw(ht + ra, auto_adjust = FALSE, ht_gap = unit(0, "mm"),

  heatmap_legend_side ="bottom",

  merge_legend = TRUE,

  row_sub_title_side = "right",

  annotation_legend_list = list(lgd_sig, lgd_sig2, lgd_sig3),

  annotation_legend_side ="bottom")

>tbl <- table(factor(sampleType$type, levels=Group))

>for(i in 1:(length(tbl)-1)){

  decorate_heatmap_body("ht", {

    grid.lines(c(sum(tbl[c(1:i)])/sum(tbl), sum(tbl[c(1:i)])/sum(tbl)), c(0, 1), gp =
gpar(lty = 2, lwd = 1))

  }, slice=1)

}

>dev.print(png, paste0(dir_singscore,"IgAN_pathways2.png"), width = 8, height = 4, uni-
ts="in", res=300)

>dev.print(pdf, paste0(dir_singscore,"IgAN_pathways2.pdf"), width = 8, height = 4)
```

Note: Based on the results of analysis of variance and Kruskal-Wallis non-parametric test, we visualize these 12 pathways with heatmap (Figure 5). The results show that IgAN-C1 and IgAN-C3 are similar in activated pathways, both of which are enriched in complement activity, extracellular matrix organization, and mitochondrial injury.

EXPECTED OUTCOMES

In this protocol, all analysis results are saved in subdirectories and output in the form of PNG and PDF files. In short, the expected results include the quality control of proteomic data, as shown in Figure 1; the consensus clustering of different k values, as shown in Figure 2; the determination of the number of disease molecular subtypes, as shown in Figures 3 and 4; and the pathway enrichment of different subtypes, as shown in Figure 5.

LIMITATIONS

This protocol currently only attempts molecular subtyping based on proteomic data. Secondly, the determination of the number of molecular subtypes cannot be automated, and still requires the interpretation of domain experts.

TROUBLESHOOTING

Problem 1

An error occurred while reading the proteome data files (step 5).

Potential solution

An error prompt "cannot open the connection" may appear when reading files, usually due to the current working directory is not clear. The best solution is to use the setwd () function to set the current working directory to the same location as the file save path.

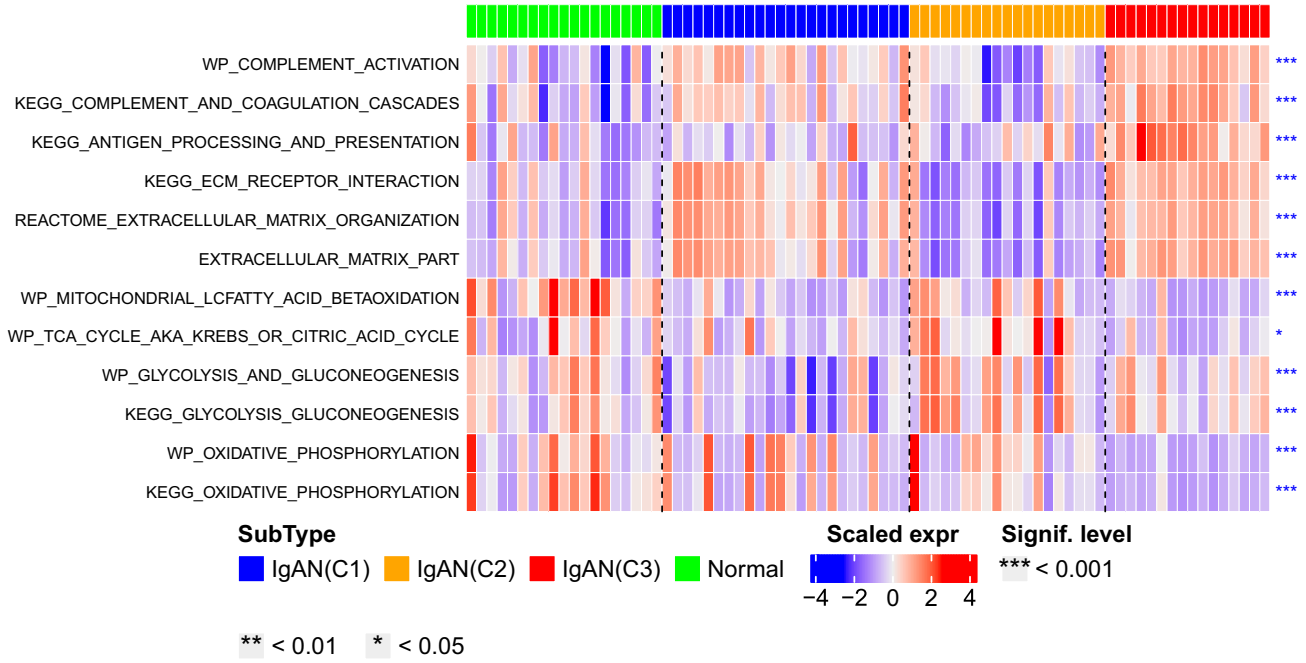


Figure 5. Heatmap of pathway's enrichment score in three IgAN subtypes and normal control kidney samples

Problem 2

The draw() function may have an error when drawing a heatmap of differentially expressed proteins (step 22c).

Potential solution

This is due to the draw() function included in both the ComplexHeatmap package and the R.utils package. When drawing a heatmap, if both packages are loaded, the detach() function can be used to return the R.utils package back to the library.

Problem 3

An error occurred while performing consensus clustering analysis on proteome data (step 16).

Potential solution

The error may be due to missing values or outliers in the data, which lead to an error in calculating the distance. Please check whether the data characteristics are accurate and whether there are missing values or outliers.

Problem 4

An error occurred during internal evaluation of different clustering results (step 18).

Potential solution

When using Davies-Bouldin's Index for clustering evaluation, the error "could not find function 'index.db'" appears. This may be because the user has not installed or loaded the R package "clusterSim", please carefully check whether the required R package has been successfully loaded before calling the function.

Problem 5

An error occurred while saving the drawn graph (step 19b).

Potential solution

This may be because multiple figures are drawn on the current graphics device, causing figure save overlap or confusion. Users can save the figures immediately after each drawing generation and use the dev.off() function to close unnecessary graphics devices.

RESOURCE AVAILABILITY

Lead contact

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Yunping Zhu (zhuyunping@gmail.com).

Technical contact

Questions about the technical specifics of performing the protocol should be directed to the technical contact, Mansheng Li (limansheng@gmail.com).

Materials availability

This study did not generate new unique reagents.

Data and code availability

This study did not generate new datasets. The datasets used for presentation in this protocol are from our previous study,[1] which are available at iProX: PXD032710 (https://www.iprox.cn). Furthermore, we have also provided a zip file 'IgAN' (https://zenodo.org/records/10512182) on Zenodo (https://zenodo.org/), which contains all the loaded data and the complete code for this protocol. Users can download the zip file directly.

ACKNOWLEDGMENTS

This work was supported by the National Key Research and Development Project (no. 2021YFA1301603 and 2018YFE0126600) and the Science and Technology Project of Beijing (D1811000001118002). The computing resources for this study were provided by the bioinformatics platform of the National Center for Protein Sciences (PHOENIX, Beijing).

AUTHOR CONTRIBUTIONS

J.W. and Y.L. tested the workflow and wrote the manuscript. X.C., M.L., and Y.Z. conceptualized the project. M.L. designed and developed the workflow and wrote the manuscript. All authors read and approved the manuscript.

DECLARATION OF INTERESTS

The authors declare no competing interests.

REFERENCES

1. Chen, X., Li, M., Zhu, S., Lu, Y., Duan, S., Wang, X., Wang, Y., Chen, P., Wu, J., Wu, D., et al. (2023). Proteomic profiling of IgA nephropathy reveals distinct molecular prognostic subtypes. iScience *26*, 105961. https://doi.org/10.1016/j.isci.2023.105961.

2. Wu, Q., Sui, X., and Tian, R. (2021). Se pu. Chinese journal of chromatography *39*, 112–117.

3. R Core Team (2022). R: A Language and Environment for Statistical Computing. https://www.r-project.org/.

4. RStudio Team (2020). RStudio: Integrated Development for R. http://www.rstudio.com/.

5. Monti, S., Tamayo, P., Mesirov, J., and Golub, T. (2003). Consensus clustering: a resampling-based method for class discovery and visualization of gene expression microarray data. Mach. Learn. *52*, 91–118. https://doi.org/10.1023/A:1023949509487.

6. Şenbabaoğlu, Y., Michailidis, G., and Li, J.Z. (2014). Critical limitations of consensus clustering in classdiscovery. Sci. Rep. *4*, 6207. https://doi.org/10.1038/srep06207.

7. Tibshirani, R., Walther, G., and Hastie, T. (2001). Estimating the Number of Clusters in a Data Set Via the Gap Statistic. JR Stat Soc Series B Stat Methodol. *63*, 411–423. http://www.jstor.org/stable/2680607.

8. John, C.R., Watson, D., Russ, D., Goldmann, K., Ehrenstein, M., Pitzalis, C., Lewis, M., and Barnes, M. (2020). M3C: Monte Carlo reference-based consensus clustering. Sci. Rep. *10*, 1816. https://doi.org/10.1038/s41598-020-58766-1.