Supporting Information

Profiling the Serum Protein Corona of Fibrillar Human Islet Amyloid Polypeptide

Emily H. Pilkington,[1,2] Ove J. R. Gustafsson,[3] Yanting Xing,[4] Juan Hernandez-Fernaud,[5] Cleidi Zampronio,[5] Aleksandr Kakinen,[1] Ava Faridi,[1] Feng Ding,[4] Paul Wilson,[1,2] Pu Chun Ke[1] and Thomas P. Davis[1,2*]*

[1]ARC Centre of Excellence in Convergent Bio-Nano Science and Technology, Monash Institute of Pharmaceutical Sciences, 381 Royal Parade, Parkville, VIC 3052, Australia

[2]Department of Chemistry, University of Warwick, Library Road, CV4 4AL, Coventry, UK

[3]ARC Centre of Excellence in Convergent Bio-Nano Science and Technology, Future Industries Institute, University of South Australia, University Boulevard, Mawson Lakes, SA 5095, Australia

[4]Department of Physics and Astronomy, Clemson University, Clemson, SC 29634, USA

[5]Warwick Proteomics Research Technology Platform, School of Life Sciences, Gibbet Hill Road, University of Warwick, CV4 7AL, Coventry, UK

Blue-Native polyacrylamide gel electrophoresis (PAGE) and analysis

Blue-Native PAGE allowed the examination of protein complex formation between IAPP amyloid fibrils (0.9 mg/mL) and FBS, which were pre-incubated (2% and 50%, in water) for 2 h. A 15 µL aliquot of each sample was mixed 1:1 with chilled Native PAGE sample buffer and then transferred to 4-15% gel (Mini-Protean TGX). Blue-Native PAGE was performed at 4 $^{\circ}$C, pH = 8.3, using sequential buffer steps: briefly, Tris/Glycine buffer was supplemented with Coomassie Brilliant Blue G-250 at either 0.02%, 0.002%, or omitted, with each buffer solution utilized in this order during the assay. For protein binding capacity a densitometry of lane and well band profiles, normalized against background intensity, was performed using ImageJ. The experiment was performed and analyzed in duplicate. All materials listed were sourced from BioRad.

Circular dichroism spectroscopy

Structural changes in bovine serum albumin (BSA), the most abundant protein species in FBS medium, were examined upon its interaction with IAPP fibrils. BSA (0.1 mg/mL) was exposed to IAPP amyloid fibrils (>1 week old, in water; 0.1 mg/mL) for 24 h, and CD spectra of BSA were obtained at different timepoints using an Aviv Model 410 CD spectrophotometer (Biomedical, Inc.). The spectra were recorded over a wavelength range of 190~260 nm, with a 1 nm step size and a scanning speed of 15 nm/min at room temperature. The final spectra were baseline-corrected and IAPP amyloid spectrum subtracted where applicable. The data were measured in mean residue ellipticity (θ) and converted to the standard units $deg \cdot cm^2\ dmol^{-1}$.

nLC-MS/MS queries, analysis and informatics

Analysis of nLC-MS/MS data utilizing custom R[1] scripts are as follows. Briefly, the combined summary files were used to plot peptide sequences identified, ratio of MS/MS identified: MS/MS submitted, the number of peaks and the mass standard deviation (ppm). Protein ID, protein name, gene name and sequence were extracted from the same uniprot reference proteome (*Bos taurus*) using *seqinr*[2] and primary sequence dependent characteristics (GRAVY, pI, MW) were calculated using *alakazam*.[3] Using protein ID as a key, the calculated values were extracted from the processed reference proteome for those proteins identified in the combined proteinGroup file to create a final data frame for plotting of identification overlaps and trends in GRAVY/pI/MW as well as amino acid composition of sequences for selected protein subsets (*seqinr, ggplot2, gridExtra, VennDiagram*).[4-6]

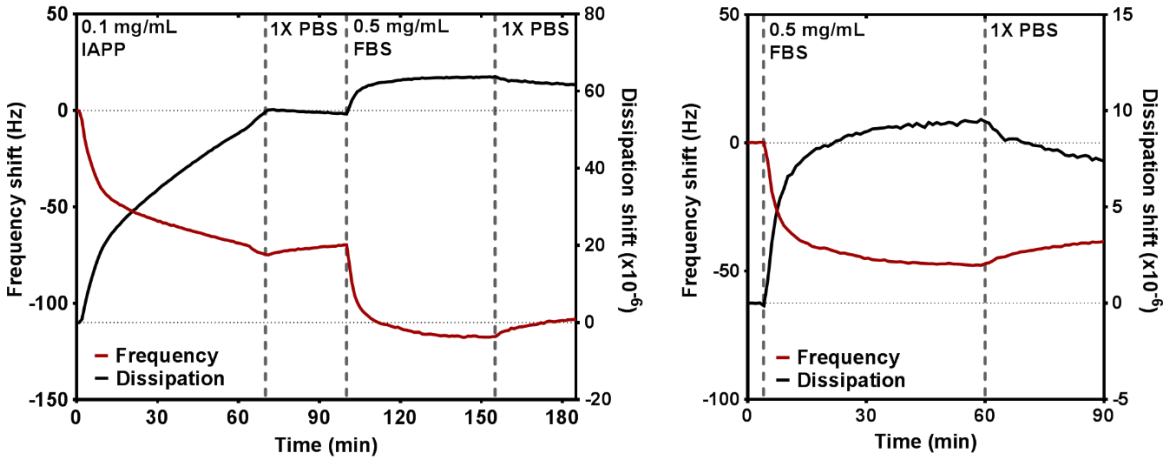


Figure S1. Protein deposition on IAPP-functionalized QCM sensors (n = 4), as illustrated by frequency and dissipation shift after sequential introduction of IAPP amyloid and FBS.

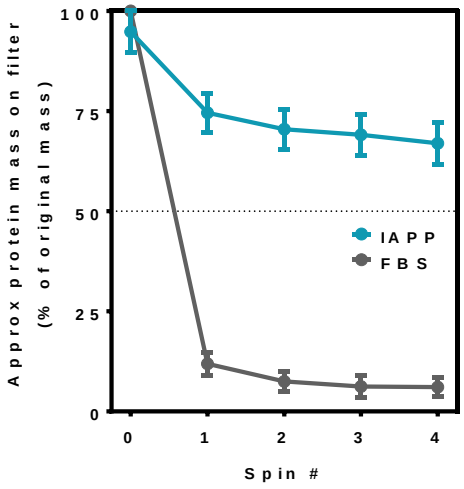


Figure S2. Optimization of centrifugal capture (CC) methodology, demonstrating retention of IAPP amyloid on high molecular weight filter surface (n = 6) with only low nonspecific binding of FBS proteins (n = 8) after four spin-wash cycles. Error is SEM.

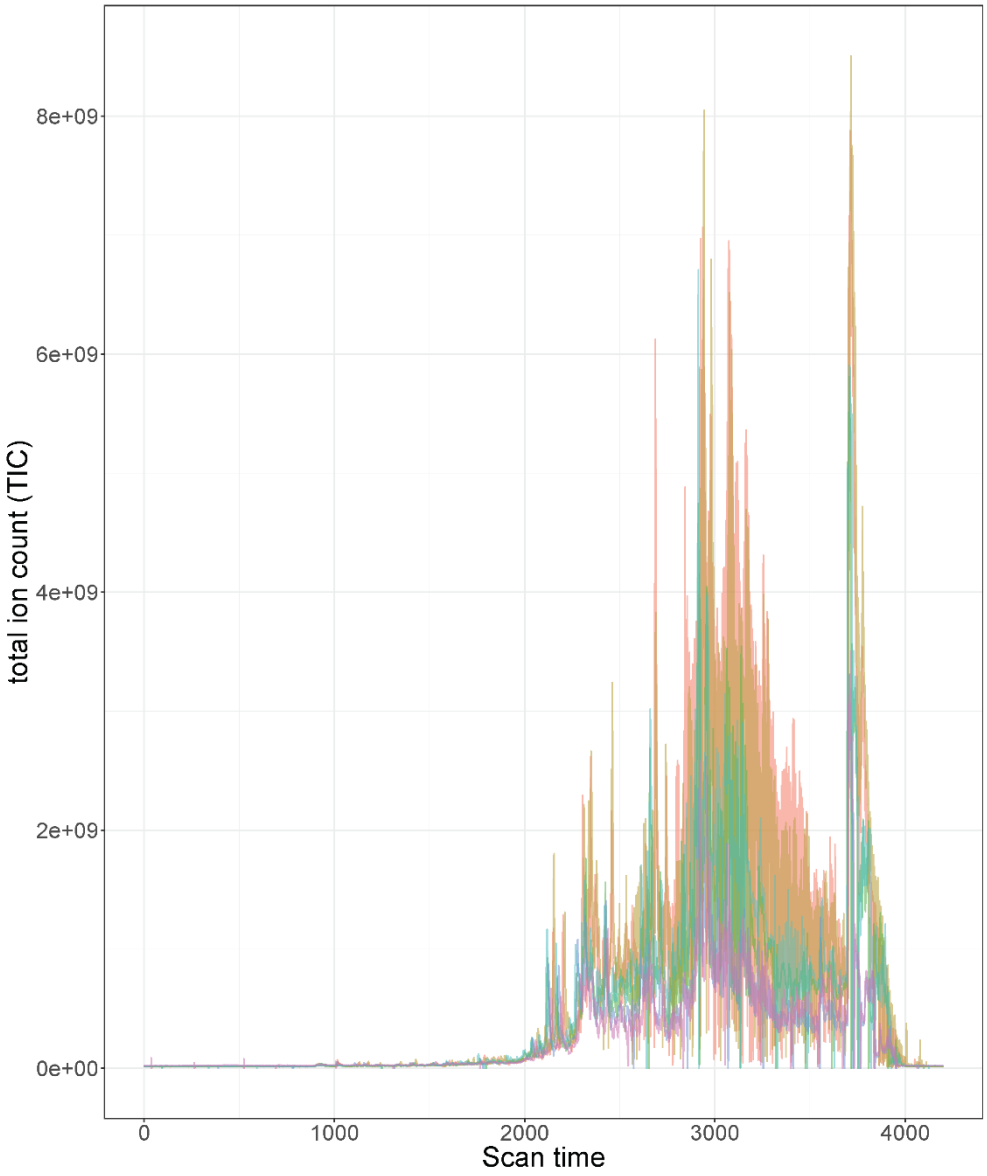


Figure S3a. LC-MS/MS chromatogram showing correlation between independent experiments of A (n = 6, overlaid).

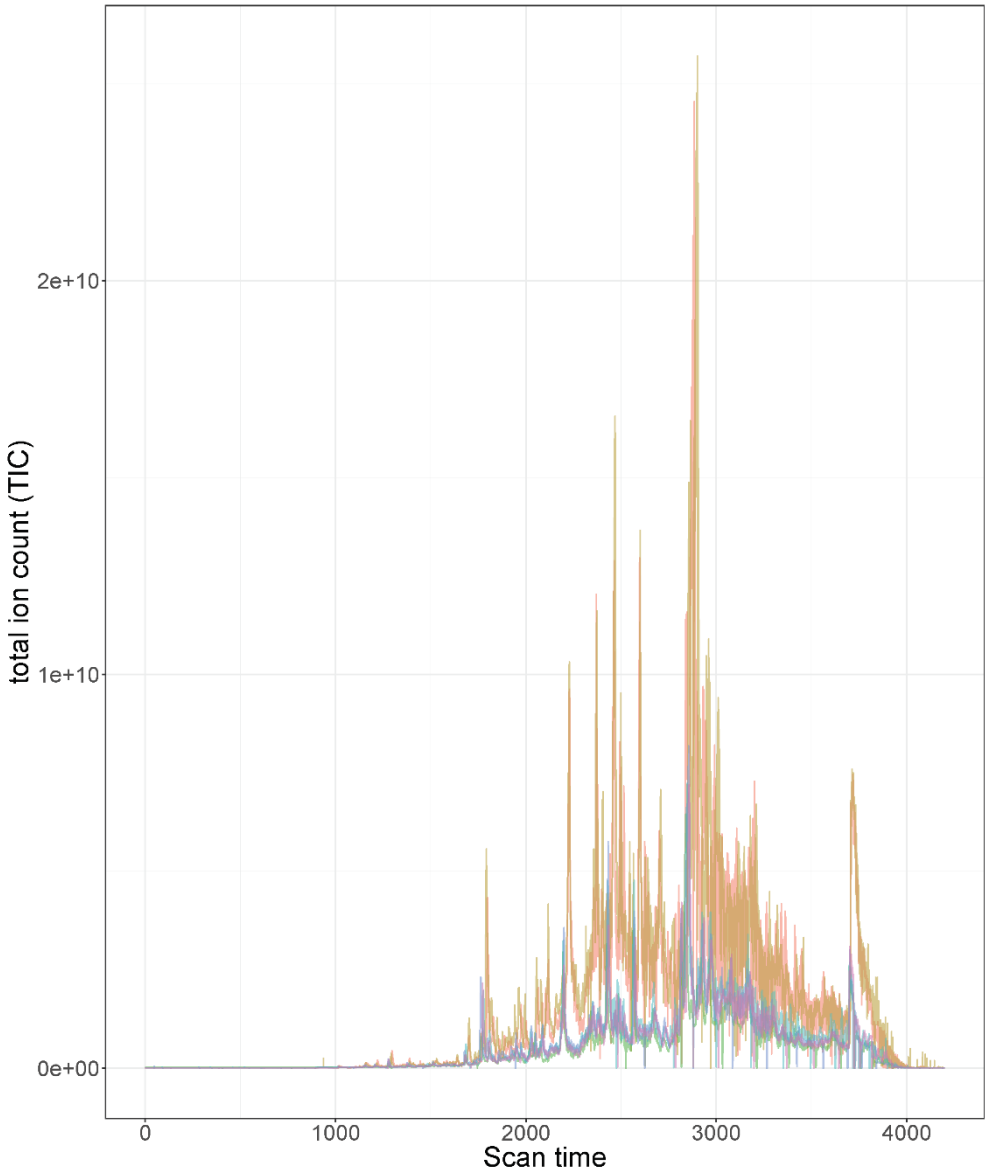


Figure S3b. LC-MS/MS chromatogram showing correlation between independent experiments of AF (n = 6, overlaid).

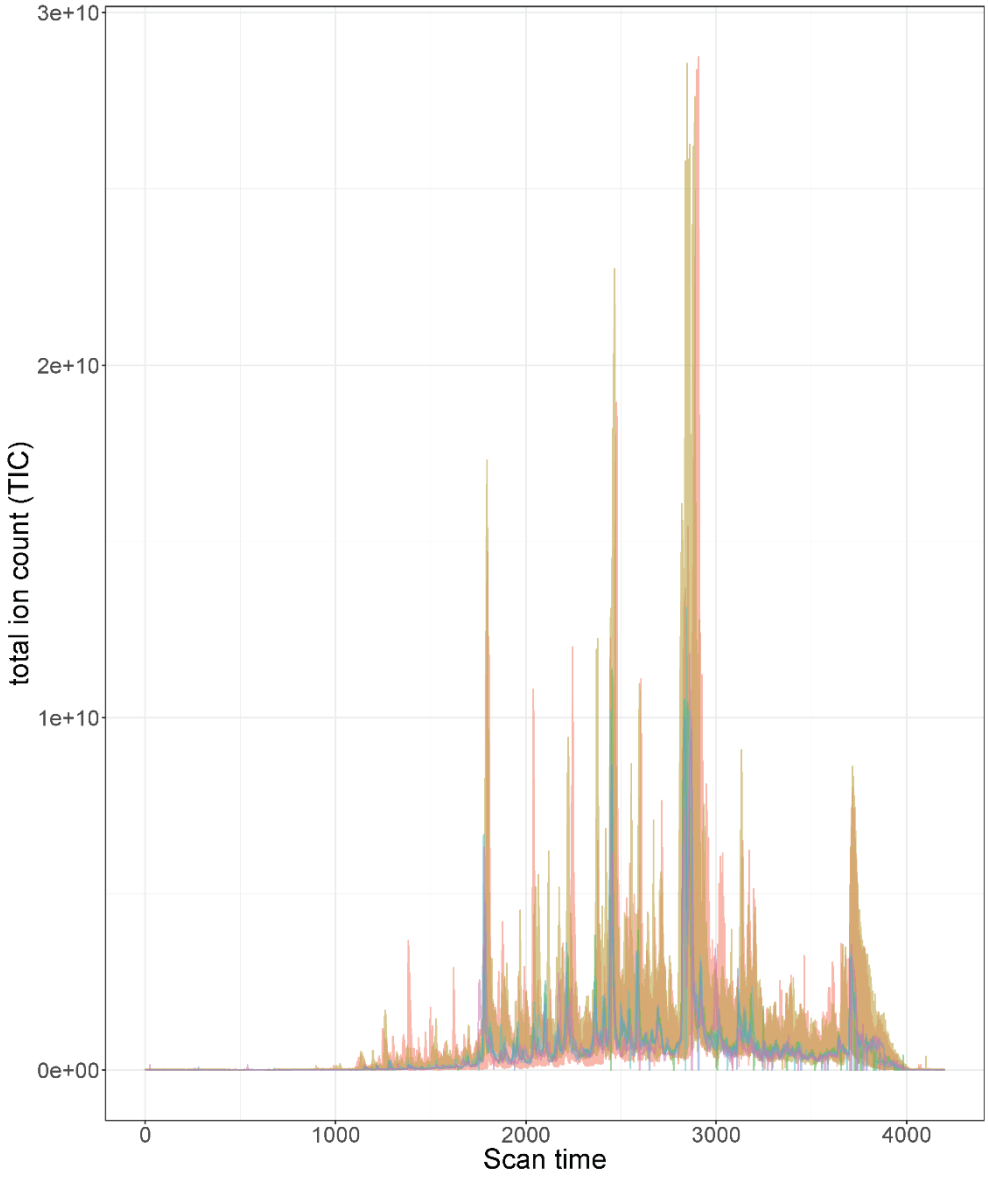


Figure S3c. LC-MS/MS chromatogram showing correlation between independent experiments of F (n = 12, overlaid).

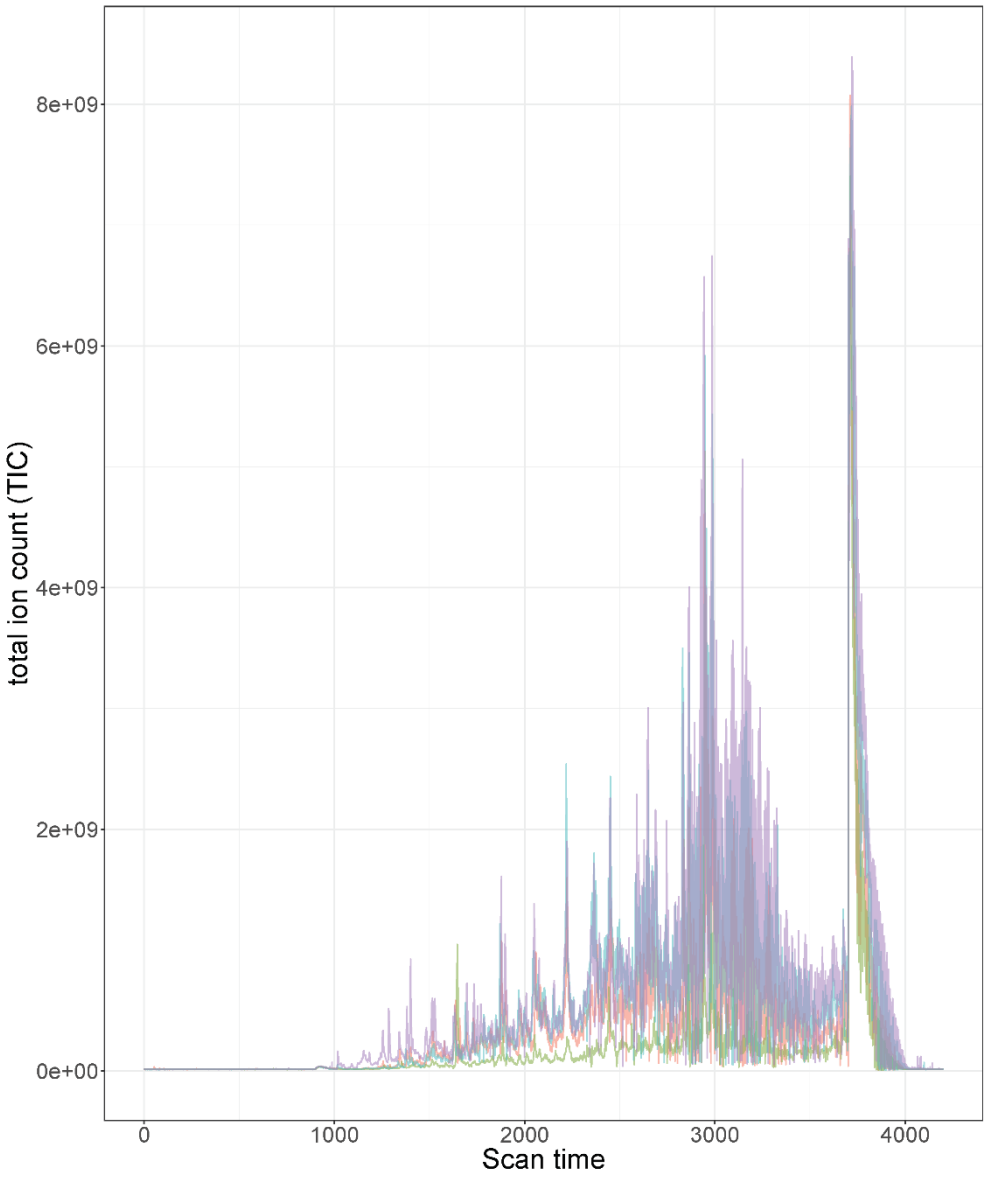


Figure S3d. LC-MS/MS chromatogram showing correlation between independent experiments of AE (n = 4, overlaid).

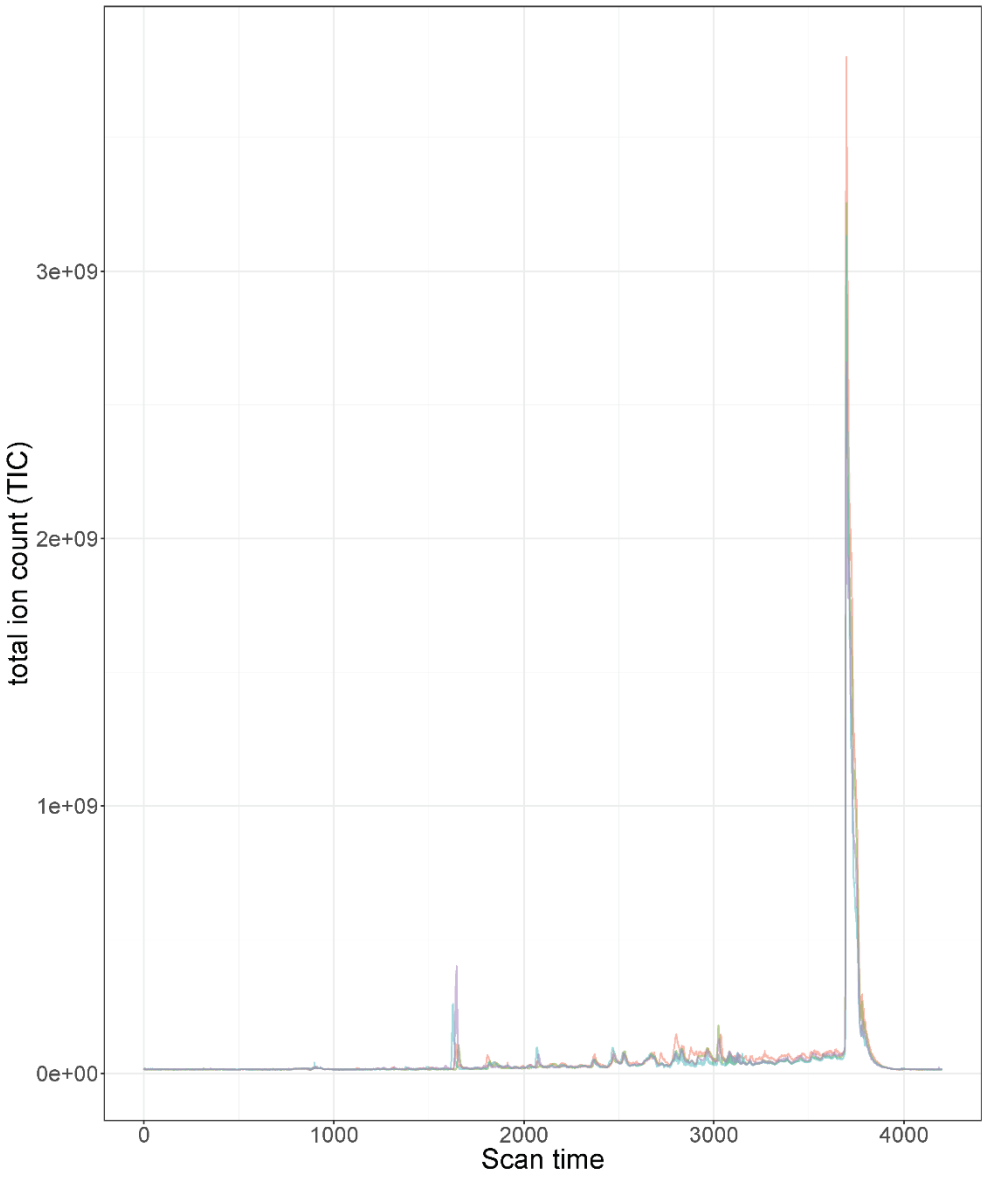


Figure S3e. LC-MS/MS chromatogram showing correlation between independent experiments of EF (n = 4, overlaid).

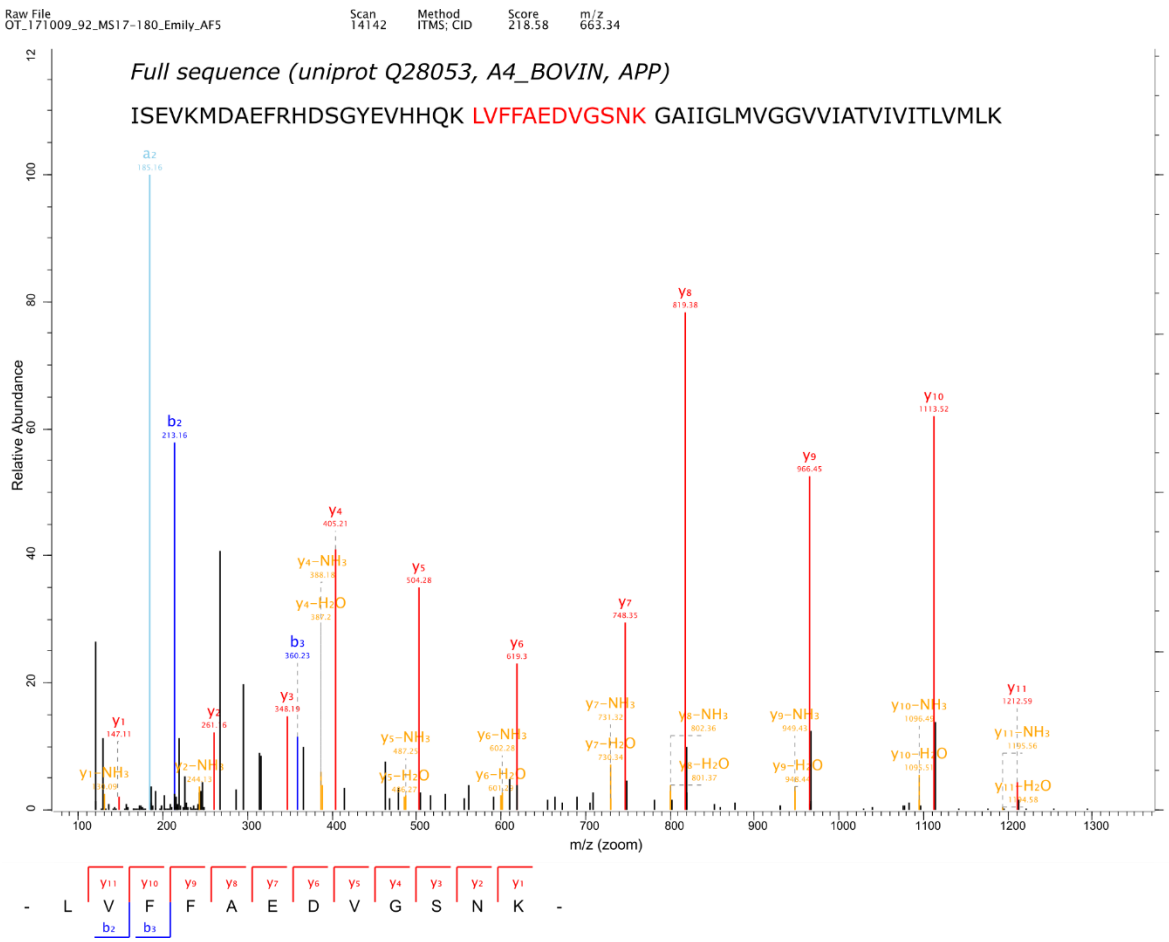


Figure S4. MS/MS spectrum identified by MaxQuant/Andromeda as a tryptic peptide of Amyloid Precursor Protein (*APP*, uniprot IDs Q28053, A4_BOVIN). Full sequence with identified peptide emphasized in red is overlaid onto the MaxQuant annotated MS/MS visualization, including matched fragments for the b and y ion series.

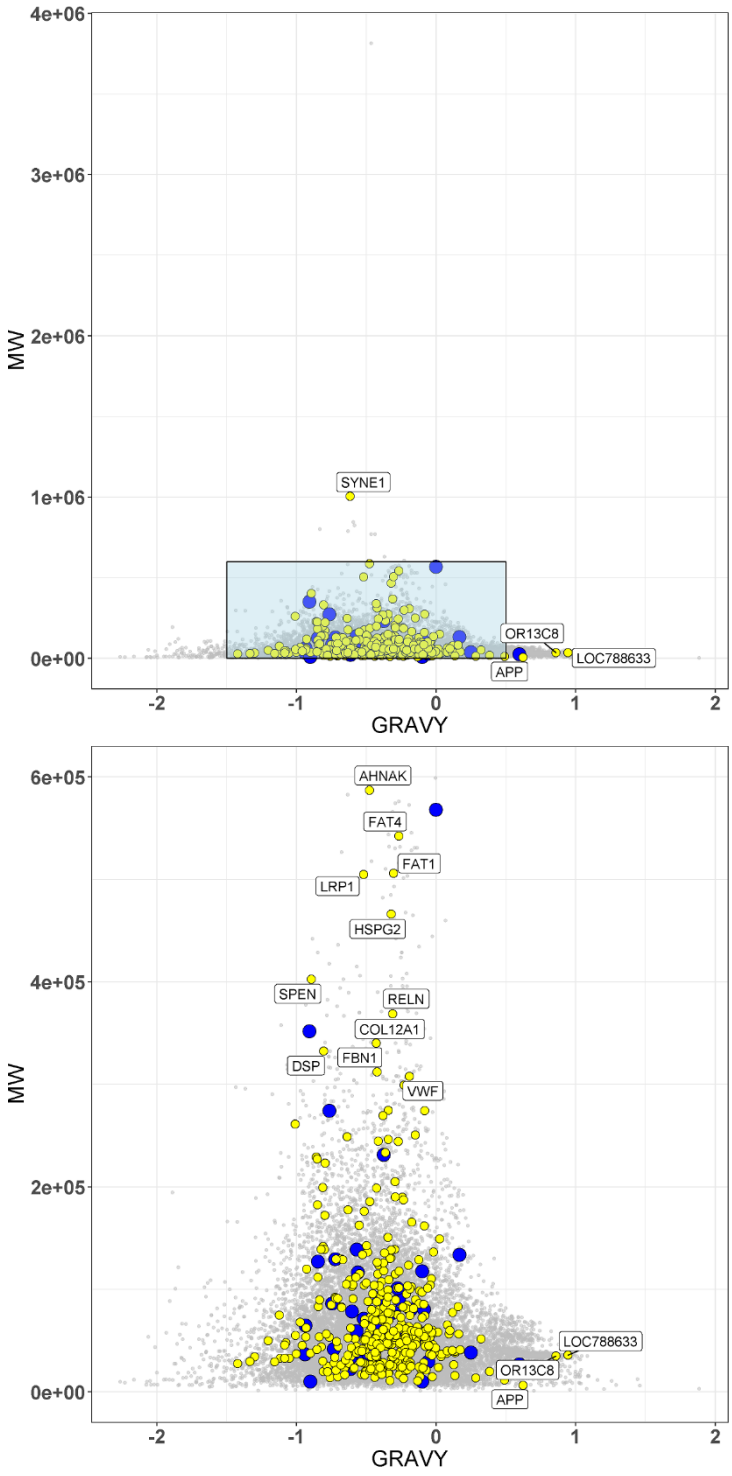


Figure S5a. The GRAVY/molecular weight (MW) relationships for unique amyloid-corona proteins (yellow markers, unique AF) and unique FBS-only proteins (blue markers, F) for CC experiments. All plots are overlaid onto the *Bos taurus* proteome background (gray). Top: points labelled are GRAVY $\leqslant$ -1.5 and $\geqslant$ 0.5, MW $\geqslant$ $6E^{+5}$. Bottom: points labelled are GRAVY (same as top) and MW $\geqslant$ $2E^{+5}$.

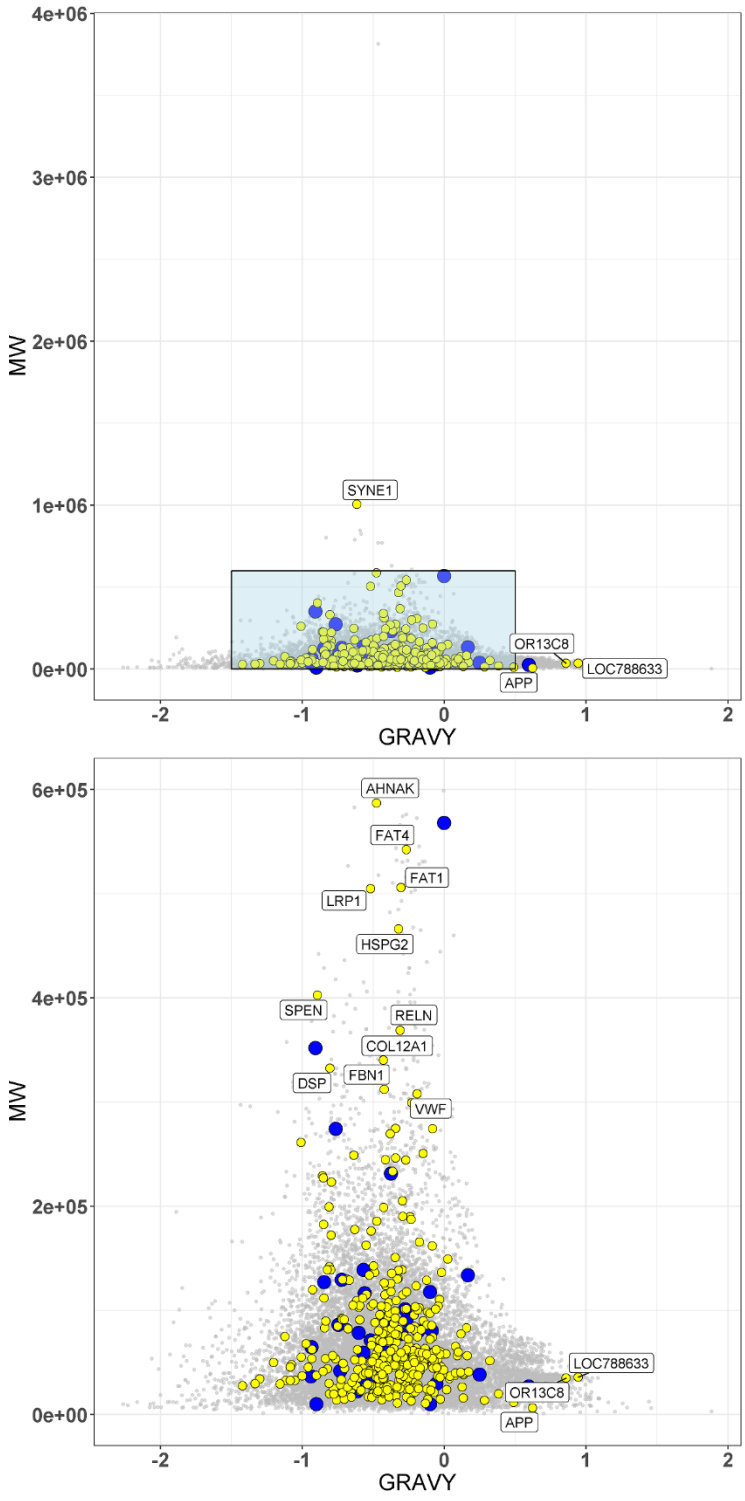


Figure S5b. The GRAVY/molecular weight (MW) relationships for unique amyloid-corona proteins (yellow markers, AE) and unique control proteins (blue markers, EF) for QCM experiments (AE/EF) are shown. Plots are overlaid onto the *Bos taurus* proteome background (gray). Top: points labelled are outside ranges GRAVY $\geqslant$ 0.5 and GRAVY $\leqslant$ -1.5, MW $\geqslant$ $6E^{+5}$. Bottom: points labelled are outside ranges GRAVY >= 0.5 and GRAVY <= -1.5, MW >= $3E^{+5}$.

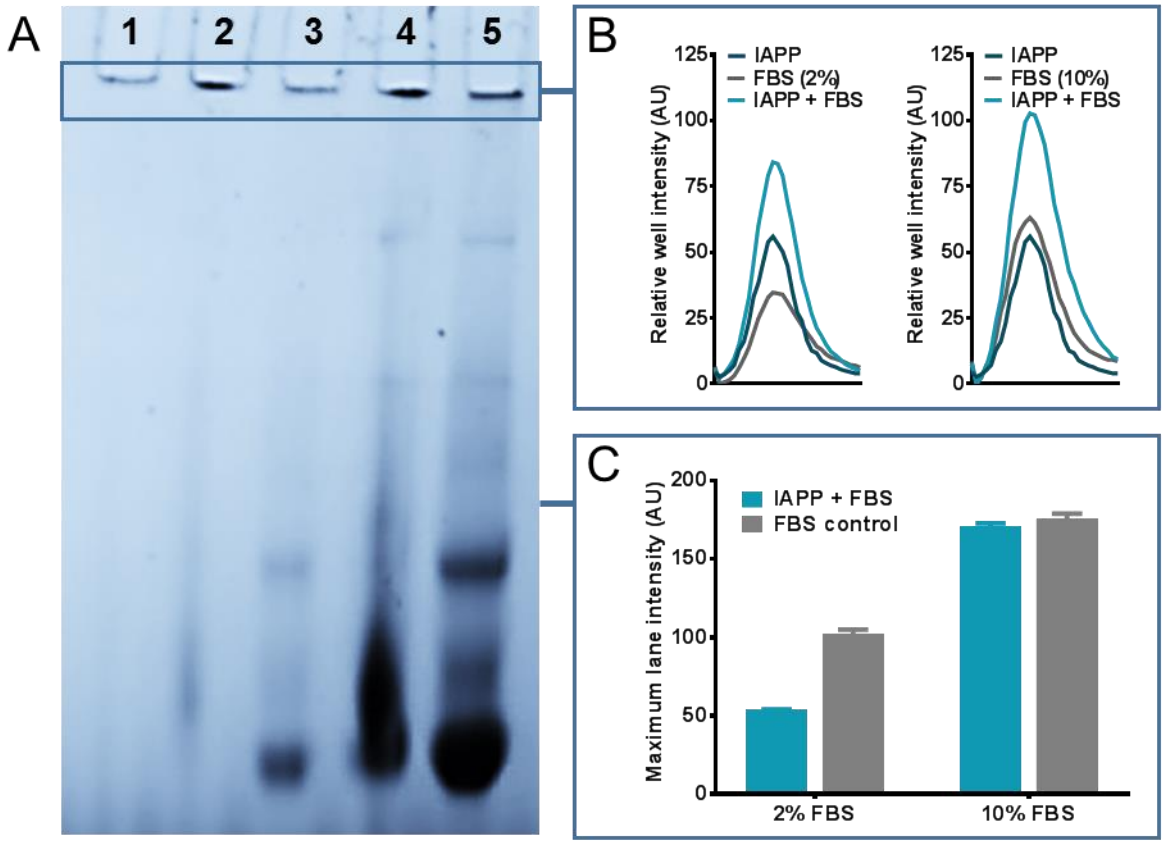


Figure S6: Blue-Native PAGE demonstrates sequestering of FBS proteins through corona formation on IAPP amyloids. A: Representative gel of n = 2 runs. Lanes are as follows: (**1**) IAPP amyloid (0.9 mg/mL); (**2**) IAPP amyloid + 2% FBS; (**3**) 2% FBS control; (**4**) IAPP amyloid + 10% FBS; (**5**) 10% FBS control. B: Intensity profile of sample wells. C: Comparison of maximum intensity measured within sample lanes (excluding well); error = SEM.

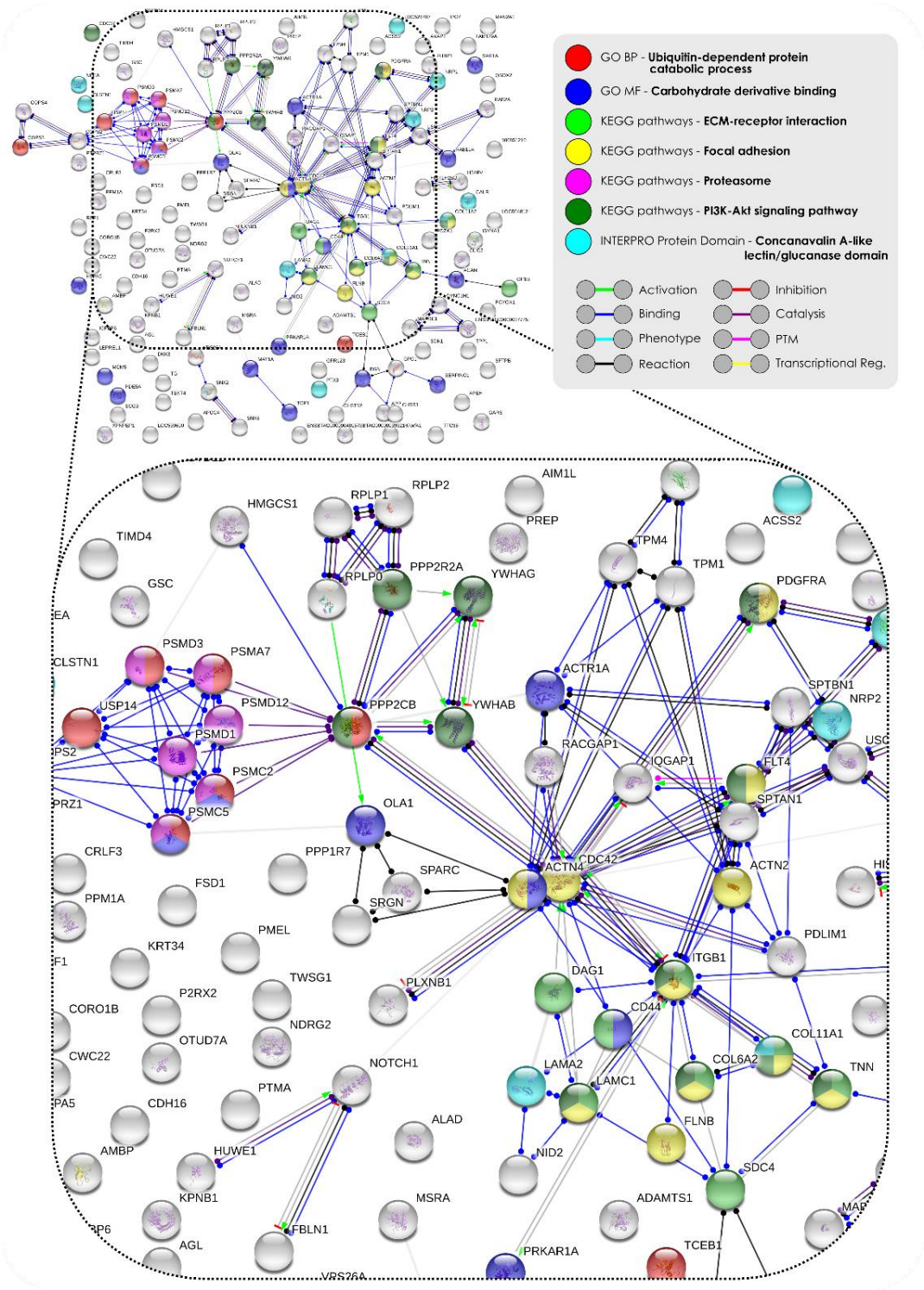


Figure S7. Centrifugal capture (CC) method amyloid and serum corona (AF) STRING (string-db.org)[7,8] protein network (molecular action) produced using *database* and *experimental* interactors, with a minimum interaction score of 0.400 and no additional interactors, against a whole *Bos taurus* genome background. Enrichment analysis and molecular action legends are included, in addition to predicted action effects – positive (arrowhead), negative (endpoint line), unspecified (endpoint circle).

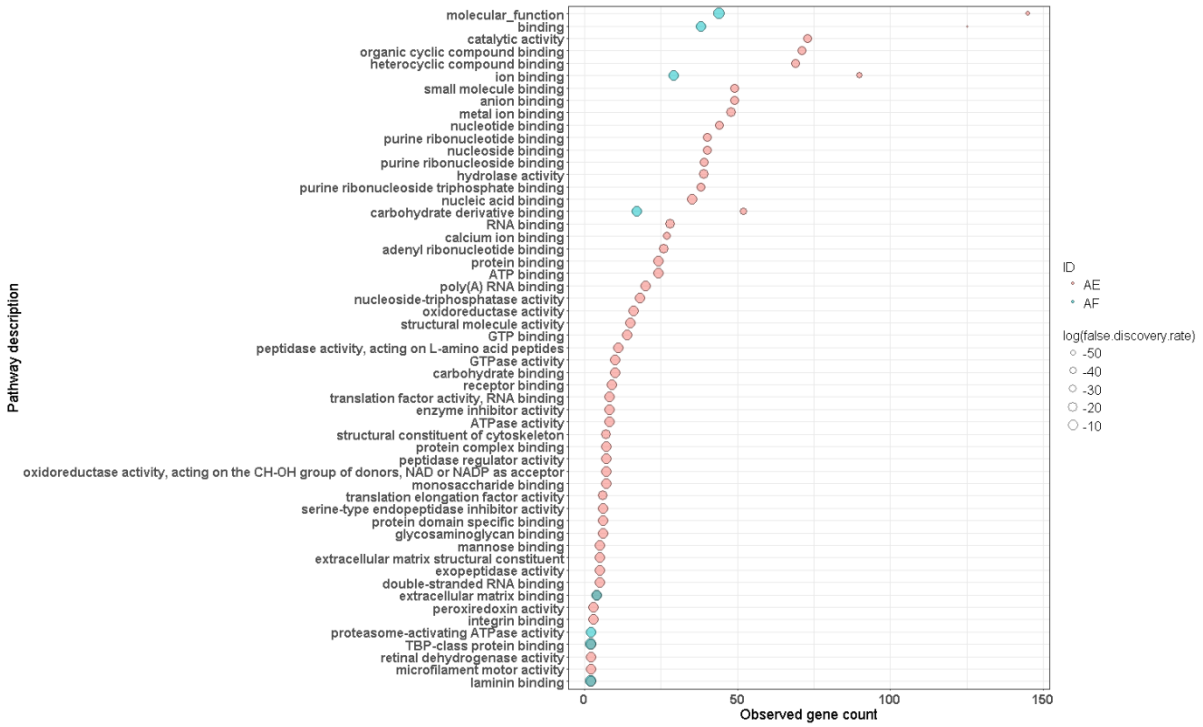


Figure S8a. Enriched gene ontology (GO) molecular function terms for proteins identified in AE (red) and AF (blue) experiments. Enrichment was performed using the STRING resource. Plot was generated in R using ggplot (see scatter plot code below). Point size is scaled to log(false.discovery.rate).

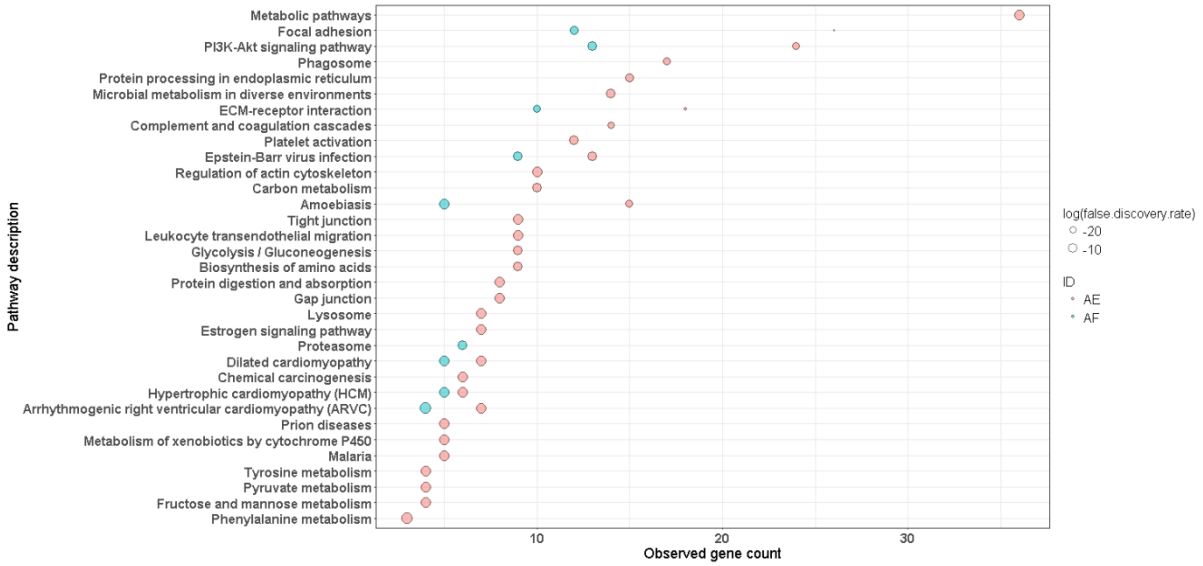


Figure S8b. Enriched gene ontology (GO) KEGG pathway terms for proteins identified in AE (red) and AF (blue) experiments. Enrichment was performed using the STRING resource. Plot was generated in R using ggplot (see scatter plot code below). Point size is scaled to log(false.discovery.rate).

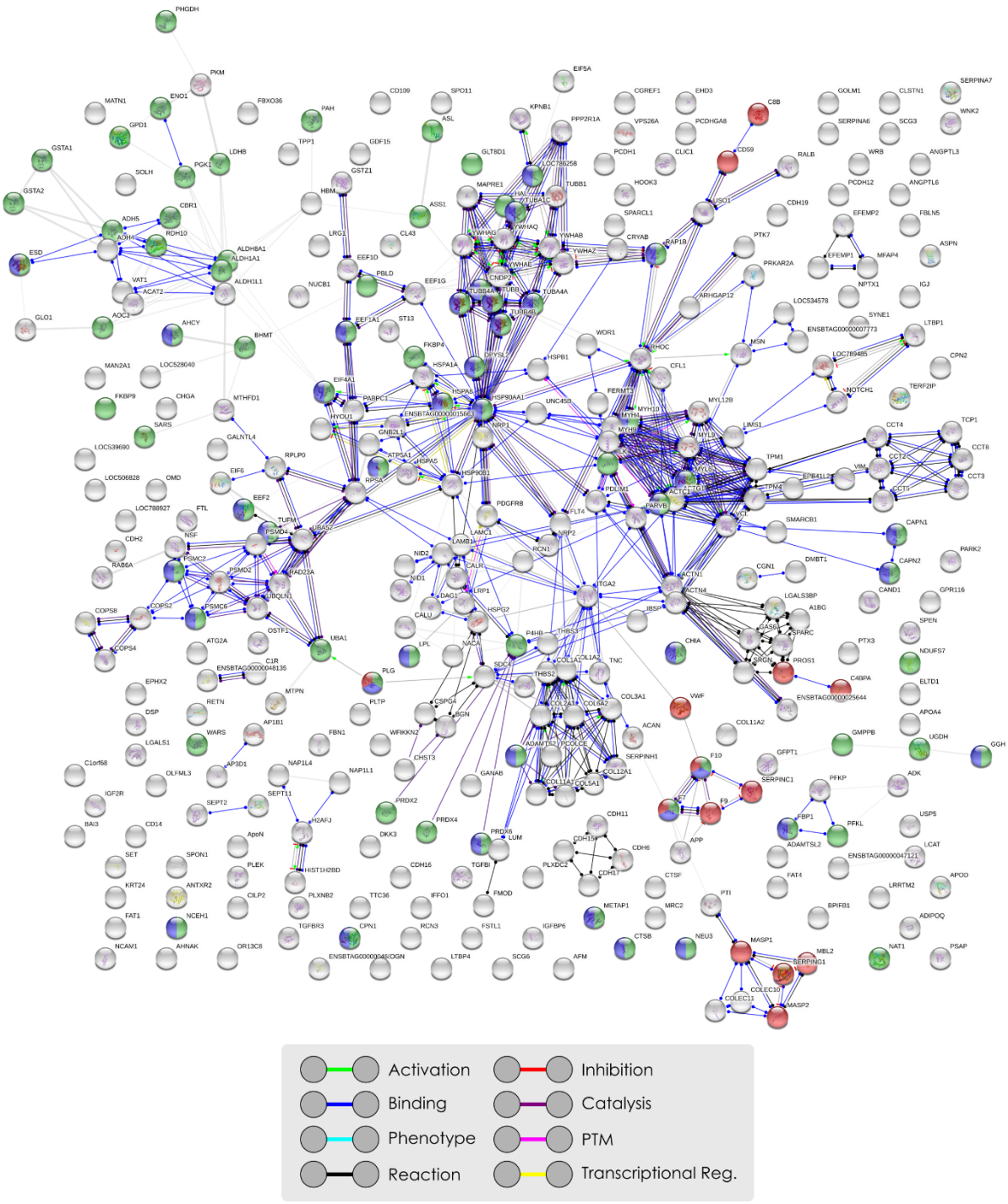


Figure S8c. Enriched gene ontology (GO) molecular functions for *Catalytic activity* (green) and *hydrolase activity* (blue), with the KEGG pathway for *Complement and coagulation cascades* also included (red). Enrichment was performed using the STRING resource. Enrichment analysis and molecular action legends are included, in addition to predicted action effects – positive (arrowhead), negative (endpoint line), unspecified (endpoint circle).

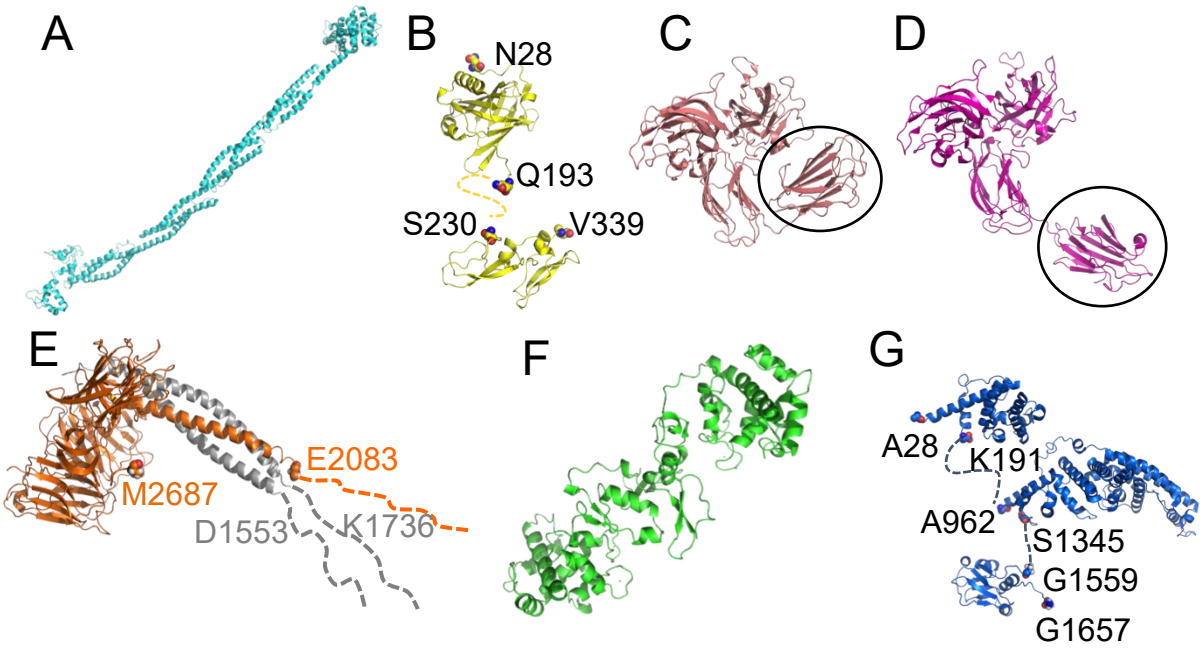


Figure S9. Structures of top AF proteins. Alpha-actinin-4 (a), protein AMBP (b), neuropilin in either close (c) or open state (d), laminin subunit alpha 2 (orange) (e), SPARC (f), and IQ motif containing GTPase activating protein 1 (g) are shown in cartoon representations. The missing structures are represented by dashed lines.

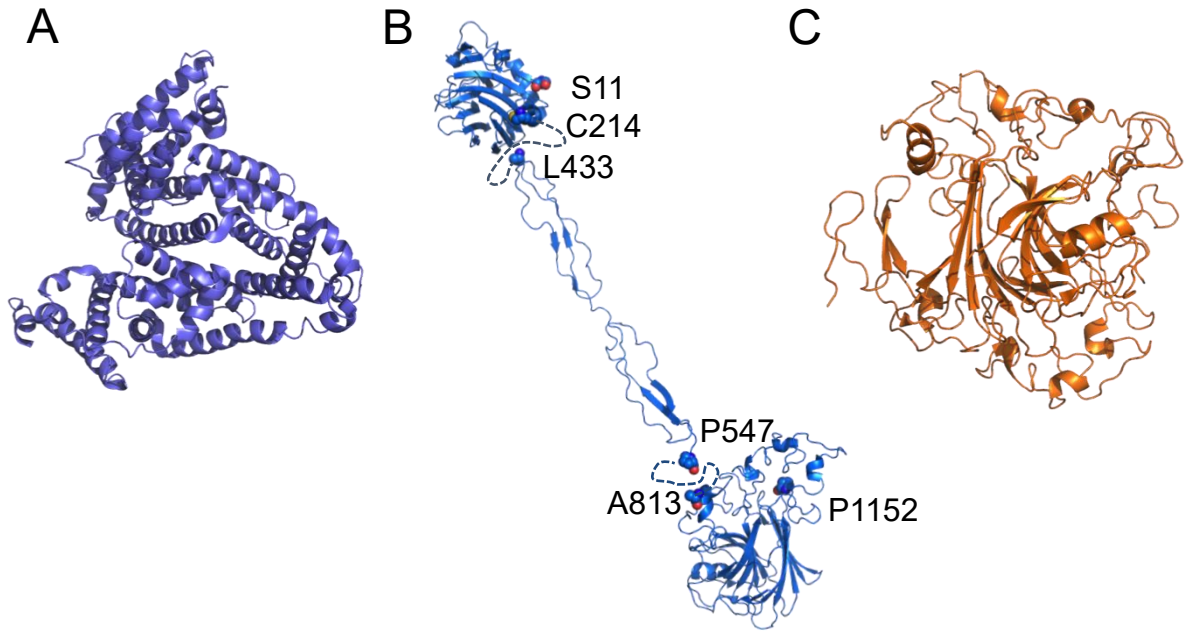


Figure S10. Structures of top AE proteins. Serum albumin (a), thrombospondin-1 (b), and cartilage oligomeric matrix protein (c) are shown in cartoon representations. The missing structures are represented by dashed lines.

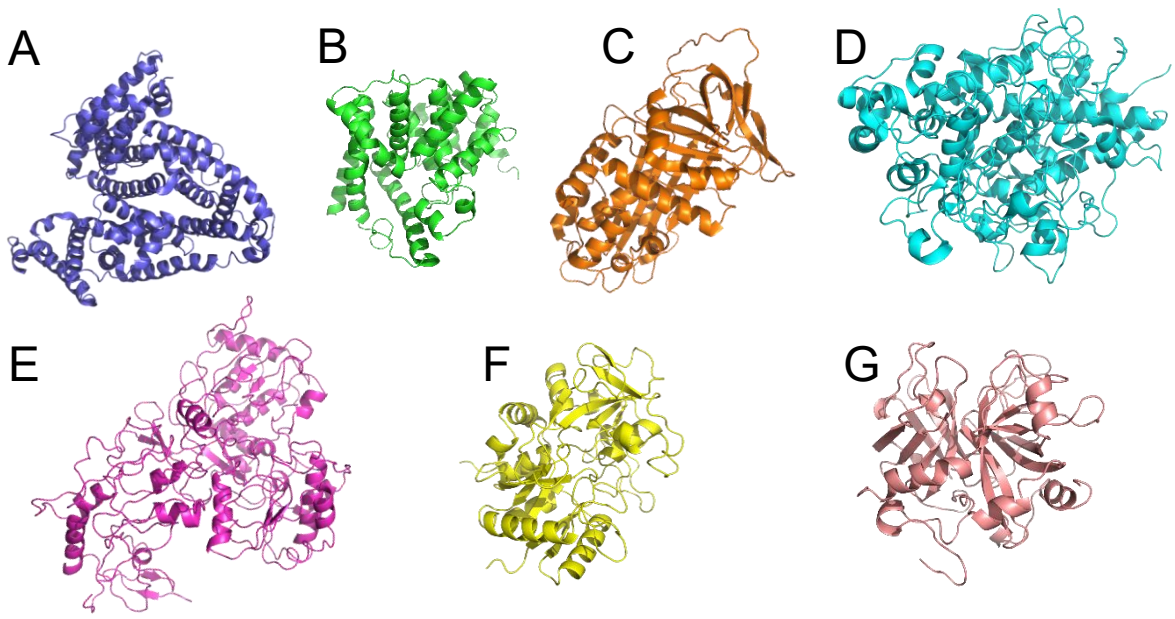


Figure S11. Structures of top FBS proteins. Serum albumin (a), cataytic domain of the Cone cGMP-specific 3′,5′-cyclic phosphodiesterase alpha-subunit (b), alpha-1-antiproteinase (c), Lactoperoxidase (d), NADH-ubiquinone oxido-reductase 75 from CroyEM (e), Hemiferrin (f), and thrombin domain of Prothrombin (g) are shown in cartoon representations.

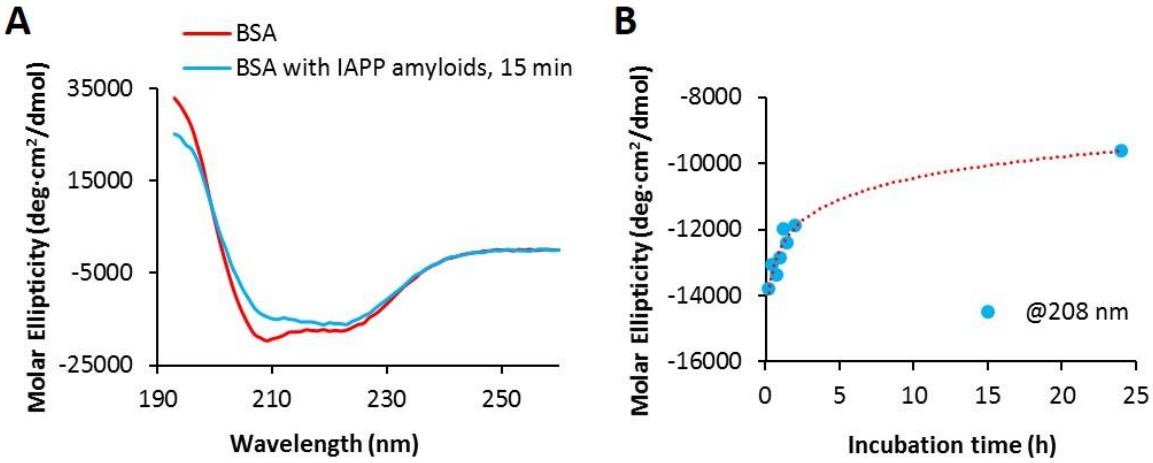


Fig S12. (A) CD spectra of BSA (0.1 mg/mL) upon its interaction with IAPP amyloid fibrils (0.1 mg/mL) after 15 min of incubation. (B) Changing intensity of CD spectra of BSA interacting with IAPP fibrils at 208 nm (one of two signature peaks of alpha helices) up to 24 h of interaction.

#	FBS protein name	PDB ID	# Residues	Net Charge
1	Serum albumin	5UJB	607	-13
2	Cone cGMP-specific 3′,5′-cyclic phosphodiesterase alpha-subunit	Catalytic domain: 3JWQ (536-855)	855	-23
3	Alpha-1-antiproteinase	1QLP (46-415) (human: 72%)	416	-9
4	Plasminogen	N/A	812	5
5	Lactoperoxidase	2IPS (118-712)	712	11
6	Kininogen, LMW II	N/A	619	-15
7	NADH-ubiquinone oxido-reductase 75	5O31 (24-727)	727	-9
8	Alpha-2-HS-glycoprotein	N/A	359	-13
9	Hemiferrin	1H76 (1-215) (rat: 70%)	216	2
10	Prothrombin	Thrombin domain: 1TBQ (318-366, 367-625)	625	-8
		Mean	594.8	-7.2

Table S1. Top 10 FBS proteins from a LC-MS method[9].

STRING scatter plot code:

```
###libraries required###
library(ggplot2)
library(ggrepel)

###theme###
#see https://rstudio-pubs-static.s3.amazonaws.com/3364_d1a578f521174152b46b19d0c83cbe7e.html
themeTIFF <- theme(legend.text = element_text(size = 16), legend.title=element_text(size=16),
axis.title=element_text(size=18),
        legend.key.size = unit(0.8, "cm"), axis.text.x = element_text(face = "bold", angle=0, vjust=0.5,
size=16),
        axis.text.y = element_text(face = "bold", size=16), title = element_text(size=18))

###molecular function###
AE <- read.csv("./input/string_input/2018_03_05_STRING_analysis/AE/enrichment.Function.tsv", sep="\t",
header=TRUE)
AF <- read.csv("./input/string_input/2018_03_05_STRING_analysis/AF/enrichment.Function.tsv", sep="\t",
header=TRUE)
###KEGG###
```

```
AE_kegg <- read.csv("./input/string_input/2018_03_05_STRING_analysis/AE/enrichment.KEGG.tsv", sep="\t",
header=TRUE)
AF_kegg <- read.csv("./input/string_input/2018_03_05_STRING_analysis/AF/enrichment.KEGG.tsv", sep="\t",
header=TRUE)

###select columns and add ID###
AE <- AE[,c(2,3,4)]
AF <- AF[,c(2,3,4)]
AE$ID <- "AE"
AF$ID <- "AF"
AE_kegg <- AE_kegg[,c(2,3,4)]
AF_kegg <- AF_kegg[,c(2,3,4)]
AE_kegg$ID <- "AE"
AF_kegg$ID <- "AF"

###combined dataframes###
COMB <- rbind(AE, AF)
COMB_kegg <- rbind(AE_kegg, AF_kegg)

###plots###
tiff("./output/molecular_function_figure.tiff", height=900, width=1500)
a <- ggplot(COMB)
a <- a + geom_point(aes(reorder(pathway.description, observed.gene.count), observed.gene.count,
                    size = log(false.discovery.rate), fill=ID), pch=21, alpha=0.5)
a <- a + theme_bw() + themeTIFF + coord_flip()
a <- a + xlab("Pathway description") + ylab("Observed gene count")
plot(a)
dev.off()

tiff("./output/kegg_figure.tiff", height=700, width=1500)
a <- ggplot(COMB_kegg)
a <- a + geom_point(aes(reorder(pathway.description, observed.gene.count), observed.gene.count,
                    size = log(false.discovery.rate), fill=ID), pch=21, alpha=0.5)
a <- a + theme_bw() + themeTIFF + coord_flip()
a <- a + xlab("Pathway description") + ylab("Observed gene count")
plot(a)
dev.off()
```

MaxQuant parameter set 1 (*Bos taurus*):

Parameter	Value
Version	1.6.0.16
User name	GUSTAFOJ
Machine name	IOR268965
Date of writing	02/19/2018 12:26:38
Fixed modifications	Carbamidomethyl (C)
Include contaminants	TRUE
PSM FDR	0.01
Protein FDR	0.01
Site FDR	0.01
Use Normalized Ratios For Occupancy	TRUE
Min. peptide Length	5

Min. score for unmodified peptides	0
Min. score for modified peptides	40
Min. delta score for unmodified peptides	0
Min. delta score for modified peptides	6
Min. unique peptides	0
Min. razor peptides	1
Min. peptides	1
Use only unmodified peptides and	TRUE
Modifications included in protein quantification	Oxidation (M);Acetyl (Protein N-term)
Peptides used for protein quantification	Razor
Discard unmodified counterpart peptides	TRUE
Label min. ratio count	2
Use delta score	FALSE
iBAQ	FALSE
iBAQ log fit	FALSE
Match between runs	FALSE
Find dependent peptides	FALSE
Fasta file	E:\MaxQuant\fasta_files\2018_02_19\UP000009136_9913.fasta
Decoy mode	revert
Include contaminants	TRUE
Advanced ratios	TRUE
Fixed andromeda index folder	
Temporary folder	
Combined folder location	
Second peptides	TRUE
Stabilize large LFQ ratios	TRUE
Separate LFQ in parameter groups	FALSE
Require MS/MS for LFQ comparisons	TRUE
Calculate peak properties	FALSE
Main search max. combinations	200
Advanced site intensities	TRUE
LFQ norm for sites and peptides	FALSE
Write msScans table	TRUE
Write msmsScans table	TRUE
Write ms3Scans table	TRUE
Write allPeptides table	TRUE
Write mzRange table	TRUE
Write pasefMsmsScans table	TRUE
Write accumulatedPasefMsmsScans table	TRUE
Max. peptide mass [Da]	4600
Min. peptide length for unspecific search	8
Max. peptide length for unspecific search	25
Razor protein FDR	TRUE
Disable MD5	FALSE
Max mods in site table	3
Match unidentified features	FALSE

MS/MS tol. (FTMS)	20 ppm
Top MS/MS peaks per Da interval. (FTMS)	12
Da interval. (FTMS)	100
MS/MS deisotoping (FTMS)	TRUE
MS/MS deisotoping tolerance (FTMS)	7
MS/MS deisotoping tolerance unit (FTMS)	ppm
MS/MS higher charges (FTMS)	TRUE
MS/MS water loss (FTMS)	TRUE
MS/MS ammonia loss (FTMS)	TRUE
MS/MS dependent losses (FTMS)	TRUE
MS/MS recalibration (FTMS)	FALSE
MS/MS tol. (ITMS)	0.5 Da
Top MS/MS peaks per Da interval. (ITMS)	8
Da interval. (ITMS)	100
MS/MS deisotoping (ITMS)	FALSE
MS/MS deisotoping tolerance (ITMS)	0.15
MS/MS deisotoping tolerance unit (ITMS)	Da
MS/MS higher charges (ITMS)	TRUE
MS/MS water loss (ITMS)	TRUE
MS/MS ammonia loss (ITMS)	TRUE
MS/MS dependent losses (ITMS)	TRUE
MS/MS recalibration (ITMS)	FALSE
MS/MS tol. (TOF)	40 ppm
Top MS/MS peaks per Da interval. (TOF)	10
Da interval. (TOF)	100
MS/MS deisotoping (TOF)	TRUE
MS/MS deisotoping tolerance (TOF)	0.01
MS/MS deisotoping tolerance unit (TOF)	Da
MS/MS higher charges (TOF)	TRUE
MS/MS water loss (TOF)	TRUE
MS/MS ammonia loss (TOF)	TRUE
MS/MS dependent losses (TOF)	TRUE
MS/MS recalibration (TOF)	FALSE
MS/MS tol. (Unknown)	0.5 Da
Top MS/MS peaks per Da interval. (Unknown)	8
Da interval. (Unknown)	100
MS/MS deisotoping (Unknown)	FALSE
MS/MS deisotoping tolerance (Unknown)	0.15
MS/MS deisotoping tolerance unit (Unknown)	Da
MS/MS higher charges (Unknown)	TRUE
MS/MS water loss (Unknown)	TRUE
MS/MS ammonia loss (Unknown)	TRUE
MS/MS dependent losses (Unknown)	TRUE
MS/MS recalibration (Unknown)	FALSE
Site tables	Oxidation (M)Sites.txt

MaxQuant parameter set 2 (*Homo sapiens*):

Parameter	Value
Version	1.6.0.16
User name	GUSTAFOJ
Machine name	IOR268965
Date of writing	02/27/2018 14:12:31
Fixed modifications	Carbamidomethyl (C)
Include contaminants	TRUE
PSM FDR	0.01
Protein FDR	0.01
Site FDR	0.01
Use Normalized Ratios For Occupancy	TRUE
Min. peptide Length	5
Min. score for unmodified peptides	0
Min. score for modified peptides	40
Min. delta score for unmodified peptides	0
Min. delta score for modified peptides	6
Min. unique peptides	0
Min. razor peptides	1
Min. peptides	1
Use only unmodified peptides and	TRUE
Modifications included in protein quantification	Oxidation (M);Acetyl (Protein N-term)
Peptides used for protein quantification	Razor
Discard unmodified counterpart peptides	TRUE
Label min. ratio count	2
Use delta score	FALSE
iBAQ	FALSE
iBAQ log fit	FALSE
Match between runs	FALSE
Find dependent peptides	FALSE
Fasta file	E:\MaxQuant\fasta_files\2018_02_23\UP000005640_9606.fasta
Decoy mode	revert
Include contaminants	TRUE
Advanced ratios	TRUE
Fixed andromeda index folder	
Temporary folder	
Combined folder location	
Second peptides	TRUE
Stabilize large LFQ ratios	TRUE
Separate LFQ in parameter groups	FALSE
Require MS/MS for LFQ comparisons	TRUE

Calculate peak properties	FALSE
Main search max. combinations	200
Advanced site intensities	TRUE
LFQ norm for sites and peptides	FALSE
Write msScans table	TRUE
Write msmsScans table	TRUE
Write ms3Scans table	TRUE
Write allPeptides table	TRUE
Write mzRange table	TRUE
Write pasefMsmsScans table	TRUE
Write accumulatedPasefMsmsScans table	TRUE
Max. peptide mass [Da]	4600
Min. peptide length for unspecific search	8
Max. peptide length for unspecific search	25
Razor protein FDR	TRUE
Disable MD5	FALSE
Max mods in site table	3
Match unidentified features	FALSE
MS/MS tol. (FTMS)	20 ppm
Top MS/MS peaks per Da interval. (FTMS)	12
Da interval. (FTMS)	100
MS/MS deisotoping (FTMS)	TRUE
MS/MS deisotoping tolerance (FTMS)	7
MS/MS deisotoping tolerance unit (FTMS)	ppm
MS/MS higher charges (FTMS)	TRUE
MS/MS water loss (FTMS)	TRUE
MS/MS ammonia loss (FTMS)	TRUE
MS/MS dependent losses (FTMS)	TRUE
MS/MS recalibration (FTMS)	FALSE
MS/MS tol. (ITMS)	0.5 Da
Top MS/MS peaks per Da interval. (ITMS)	8
Da interval. (ITMS)	100
MS/MS deisotoping (ITMS)	FALSE
MS/MS deisotoping tolerance (ITMS)	0.15
MS/MS deisotoping tolerance unit (ITMS)	Da
MS/MS higher charges (ITMS)	TRUE
MS/MS water loss (ITMS)	TRUE
MS/MS ammonia loss (ITMS)	TRUE
MS/MS dependent losses (ITMS)	TRUE
MS/MS recalibration (ITMS)	FALSE
MS/MS tol. (TOF)	40 ppm
Top MS/MS peaks per Da interval. (TOF)	10
Da interval. (TOF)	100
MS/MS deisotoping (TOF)	TRUE
MS/MS deisotoping tolerance (TOF)	0.01
MS/MS deisotoping tolerance unit (TOF)	Da

MS/MS higher charges (TOF)	TRUE
MS/MS water loss (TOF)	TRUE
MS/MS ammonia loss (TOF)	TRUE
MS/MS dependent losses (TOF)	TRUE
MS/MS recalibration (TOF)	FALSE
MS/MS tol. (Unknown)	0.5 Da
Top MS/MS peaks per Da interval. (Unknown)	8
Da interval. (Unknown)	100
MS/MS deisotoping (Unknown)	FALSE
MS/MS deisotoping tolerance (Unknown)	0.15
MS/MS deisotoping tolerance unit (Unknown)	Da
MS/MS higher charges (Unknown)	TRUE
MS/MS water loss (Unknown)	TRUE
MS/MS ammonia loss (Unknown)	TRUE
MS/MS dependent losses (Unknown)	TRUE
MS/MS recalibration (Unknown)	FALSE
Site tables	Oxidation (M)Sites.txt

Complete script used to generate Rmarkdown html report:

```
---
title: "Supplementary data analysis methods - Profiling the Serum Protein Corona of Human IAPP Amyloid"
output:
  html_document:
    toc: true
    toc_depth: 2
    toc_float: TRUE
    self_contained: no
    number_sections: TRUE
---

<style type="text/css">
#TOC {
width: 900px;
overflow:auto;
color:blue;
font-family:Helvetica;
}
body{
font-family:Helvetica;
font-size: 12px;
max-width: 2000px;
margin: auto;
margin-left:50px;
line-height: 20px;
}
h1.title{font-size: 26px}
h1{font-size: 22px}
h2{font-size: 20px}
h3{font-size: 20px}
h4{font-size: 18px}
code.r{font-size: 11px;}
```

pre{font-size: 11px;}
</style>

***

```{r setup, include=FALSE}
###global settings###
knitr::opts_chunk$set(echo = TRUE, cache=TRUE, autodep=TRUE)
###installs###
#install.packages(c("ggplot2", "gridExtra",
#                   "seqinr", "alakazam",
#                   "sqldf", "stringr",
#                   "ggrepel", "VennDiagram",
#                   "rmarkdown", "knitr"))
###libraries###
library(gridExtra)
library(ggplot2)
library(seqinr)
library(alakazam)
library(sqldf)
library(stringr)
library(ggrepel)
library(VennDiagram)
library(reshape2)
#update.packages()
```

***

Introduction

The specific research questions addressed in this document include:

1. Which proteins can be identified in IAPP amyloid-coronae formed in-solution (CC method)?
2. Which proteins can be identified in IAPP amyloid-coronae formed under micro-flow conditions (QCM method)?
3. What is the difference in protein corona composition for CC and QCM methods?
4. What proteins are identified for IAPP amyloid-only experiments?

This document details the combination, processing and presentation of summary and proteinGroup text files resulting from MaxQuant Andromeda database queries, where each experiment type was pooled to maximise identifications (e.g. A1+A2+A3....).
Independent searches were performed for selected experiment types (e.g. AF, F).

***

Analysis notes

General information

- MaxQuant peptide identifications used a fasta file containing all reviewed and unreviewed Bos taurus entries available as a reference proteome @ ftp://ftp.uniprot.org/pub/databases/uniprot/current_release/knowledgebase/reference_proteomes/Eukaryota/.
- Reviewed + unreviewed entries included to maximise identifications.
- Database 1: UP000009136_9913.fasta dated 21/12/2017.
- Database 2: UP000005640_9606.fasta dated 21/12/2017.
- Whole proteome of Bos taurus as plotting background (same fasta as above, 9136_9913).
- nLC-MS chromatograms imported from a separate R script (provided at the end of this document).
- Protein network analysis images were imported from a manual analysis using the online resource STRING (v10.5).
- STRING resource link - see https://string-db.org/cgi/input.pl

Plotting theme

```{r}
###set theme###
#see https://rstudio-pubs-static.s3.amazonaws.com/3364_d1a578f521174152b46b19d0c83cbe7e.html
```

```
themeTIFF <- theme(legend.text = element_text(size = 16), legend.title=element_text(size=16),
axis.title=element_text(size=18),
              legend.key.size = unit(0.8, "cm"), axis.text.x = element_text(face = "bold", angle=0, vjust=0.5, size=16),
              axis.text.y = element_text(face = "bold", size=16), title = element_text(size=18))
```

***

Experiment key

ID	Method	Meaning
A	CC	Control - Amyloid only
F	CC	Control - FBS only
AF	CC	Amyloid + FBS
EF	QCM	Control - FBS only
AE	QCM	Amyloid + FBS

***

Combined MaxQuant files

Summary files

Function to combine MaxQuant summary files for analysis of the nLC-MS/MS data.

This analysis includes:

- Peptide sequences identified,
- Ratio of MS/MS identified : MS/MS submitted,
- The number of peaks, and
- The mass standard deviation (ppm)

```{r sum}
###set path###
path = "./input/IAPP"

###GETsummaries###
GETsummaries <- function(x){
  #see https://stackoverflow.com/questions/4876813/using-r-to-list-all-files-with-a-specified-extension
  sum_files <- list.files(path = x, pattern = "summary.txt", recursive=TRUE, include.dirs=TRUE, ignore.case =
TRUE)
  complete_summary <- data.frame()
  for (i in sum_files){
    #see https://stackoverflow.com/questions/8996134/extract-vectors-from-strsplit-list-without-using-a-loop
    Experiment <- rapply(strsplit(i, "/"), function(a) a[1])
    summary <- read.csv(paste(x, "/", i, sep=""), sep="\t", header=TRUE)
    summary <- summary[summary$Raw.file!="Total",]
    summary$Experiment <- Experiment
    id <- rapply(strsplit(as.character(summary$Raw.file), "_"), function(a) a[4])
    exp <- rapply(strsplit(as.character(summary$Raw.file), "_"), function(a) a[6])
    ID_EXP <- paste(id, exp, sep="-")
    summary$ID_EXP <- ID_EXP
    complete_summary <- rbind(summary, complete_summary)
    }
  return(complete_summary)
}
summaries <- GETsummaries(path)
```

Summary file plot

```{r sumplot}
tiff(file=paste("./output/summary_plots.tiff", sep=""), width=600, height=1000)
#see https://www.statmethods.net/advgraphs/layout.html
a <- ggplot(summaries)
b <- a + geom_bar(stat="identity", aes(ID_EXP, Peptide.Sequences.Identified, fill=Experiment))
```

```
b <- b + expand_limits(y=0) + coord_flip() + xlab("Experimental ID") + theme_bw()
#see http://www.cookbook-r.com/Graphs/Legends_(ggplot2)/
b <- b + guides(fill=FALSE)
c <- a + geom_bar(stat="identity", aes(ID_EXP, MS.MS.Identified..../MS.MS.Submitted, fill=Experiment))
c <- c + expand_limits(y=0) + coord_flip() + xlab("Experimental ID") + theme_bw() + guides(fill=FALSE)
d <- a + geom_bar(stat="identity", aes(ID_EXP, Mass.Standard.Deviation..ppm., fill=Experiment))
d <- d + expand_limits(y=0) + coord_flip() + xlab("Experimental ID") + theme_bw()
e <- a + geom_bar(stat="identity", aes(ID_EXP, Peaks, fill=Experiment))
e <- e + expand_limits(y=0) + coord_flip() + xlab("Experimental ID") + theme_bw() + guides(fill=FALSE)
grid.arrange(b, c, e, d, ncol=2, nrow=2)
dev.off()
```

The number of peptide sequences identified (top left), ratio of MS/MS identified : MS/MS submitted (top right), the number of peaks (bottom left) and the mass standard deviation (ppm, bottom right) for the indicated shotgun nLC-MS/MS experiments.

proteinGroup files
Function to combine MaxQuant outputs for subsequent analyses.
The Experiment ID (see Experiment key) is added to the dataframe here to allow subsetting of the data.

```{r group}
GETgroups <- function(x){
  #see https://stackoverflow.com/questions/4876813/using-r-to-list-all-files-with-a-specified-extension
  prot_files <- list.files(path = x, pattern = "proteinGroups.txt", recursive=TRUE, include.dirs=TRUE,
ignore.case=TRUE)
  complete_group <- data.frame()
  count <- 0
  for (i in prot_files){
    #see https://stackoverflow.com/questions/8996134/extract-vectors-from-strsplit-list-without-using-a-loop
    Experiment <- rapply(strsplit(i, "/"), function(a) a[1])
    prot_group <- read.csv(paste(x, "/", i, sep=""), sep="\t", header=TRUE)
    prot_group$Experiment <- Experiment
    count <- count + nrow(prot_group)
    print(paste("Experiment ", Experiment, " identified ", count, " proteins", sep=""))
    count <- 0
    complete_group <- rbind(prot_group, complete_group)
  }
  return(complete_group)
}
protgroups <- GETgroups(path)
```

The unique Experiments included in the final proteinGroups data frame are `r unique(protgroups$Experiment)`.
The total identifications for each appear in the table below.

```{r}
###count Experiment-specific ID###
table(protgroups$Experiment)
```

Uniprot database

Load database

Here, the Bos taurus uniprot database file "UP000009136_9913.fasta"" is loaded and the protein ID, protein name, gene name and sequence are extracted.

```{r up}
###read uniprot fasta file using seqinr function read.fasta###
UP <- read.fasta(file="./input/fasta/UP000009136_9913.fasta", seqtype="AA", as.string=TRUE)

###function to extract sequences and annotations from fasta files###
```

```
GetAnnotSeqFASTA <- function(x){
  UP_sequence <- unlist(getSequence(x, as.string=TRUE))
  UP_annotations <- unlist(getAnnot(x, as.string=TRUE))
  ProtName <- rapply(strsplit(UP_annotations, "IN ", fixed=TRUE), function(a) a[2])
  UP_prot_name <- rapply(strsplit(ProtName, " OS", fixed=TRUE), function(a) a[1])
  Gene <- rapply(strsplit(UP_annotations, "GN=", fixed=TRUE), function(a) a[2])
  UP_gene <- rapply(strsplit(Gene, " ", fixed=TRUE), function(a) a[1])
  UP_prot_ID <- rapply(strsplit(UP_annotations, "|", fixed=TRUE), function(a) a[2])
  return(data.frame(UP_prot_ID, UP_prot_name, UP_gene, UP_sequence))
}

###apply function to extract sequences and annotations###
dataUP <- GetAnnotSeqFASTA(UP)

str(dataUP)
```

This uniprot Bos taurus reference proteome contains `r nrow(dataUP)` proteins.

Filter database
These are the amino acid counts across the proteome:

```{r upfilter}
#see https://stackoverflow.com/questions/19476210/counting-the-number-of-each-letter-in-a-vector-of-strings
table(unlist(strsplit(as.character(dataUP$UP_sequence), ""), use.names=FALSE))
```

To allow the package functions to work properly when calculating sequence dependent values - isoelectric point (pI), molecular weight (MW) and Grand Average of Hydropathy (GRAVY) - U, B and X residues need to be removed.

```{r}
###remove U, B and X containing sequences###
#see https://stackoverflow.com/questions/6650510/remove-rows-from-data-frame-where-a-row-match-a-string
dataUP <- dataUP[!(str_count(as.character(dataUP$UP_sequence), "U")>=1),]
dataUP <- dataUP[!(str_count(as.character(dataUP$UP_sequence), "B")>=1),]
dataUP <- dataUP[!(str_count(as.character(dataUP$UP_sequence), "X")>=1),]
```

Following removal of U, B and X containing sequences the Uniprot database contains `r nrow(dataUP)` proteins.

Calculate protein characteristics
Now the pI, MW and GRAVY can be calculated.

```{r}
###string split uniprot sequences and calculate GRAVY, pI and MW###
split_SEQ <- strsplit(as.character(dataUP$UP_sequence), "")
dataUP$pI <- unlist(lapply(split_SEQ, computePI))
dataUP$MW <- unlist(lapply(split_SEQ, pmw))
dataUP$GRAVY <- gravy(dataUP$UP_sequence)
```

Protein filtering and ID matching
Here, entries with greater than, or equal to, 1 unique peptide are retained. In addition, the first protein ID in the "Protein.IDs"" column is extracted and used from here on.

```{r extract}
###Proteins with >= 1 unique peptide retained###
protgroupsMOD <- protgroups[protgroups$Unique.peptides >= 1,]
###Multiple entries for protein IDs removed - first entry retained###
protgroupsMOD$Prot_ID <- rapply(strsplit(as.character(protgroupsMOD$Protein.IDs),";"), function(a) a[1])
```

The extracted protein ID in the combined MaxQuant protein groups summary is then used to match to the uniprot dataframe protein ID and extract the GRAVY, pI and MW values for these entries.

```{r}
###EXTRACT by matching Prot_ID in "protgroupsMOD" dataframe to UP_prot_ID from "dataUP"###
#see https://www.r-bloggers.com/manipulating-data-frames-using-sqldf-a-brief-overview/
extract <- sqldf("select * from dataUP inner join protgroupsMOD on protgroupsMOD.Prot_ID =
dataUP.UP_prot_ID")
```

Final count of protein IDs:

```{r}
###count Experiment-specific ID###
table(extract$Experiment)
```

Data subsets
These data subsets are used for the plots generated by this script.

```{r subsets}
###combined experiment subsets###
Q_AF_F <- extract[extract$Exp=="AF" | extract$Exp=="F",]
Q_AE_EF <- extract[extract$Experiment=="AE" | extract$Experiment=="EF",]
Q_AF_AE <- extract[extract$Exp=="AF" | extract$Exp=="AE",]
Q_AF_AE_EF_F <- extract[extract$Experiment=="AF" | extract$Experiment=="AE" | extract$Experiment=="EF" |
extract$Experiment=="F",]

###single experiment subsets###
Q_A <- extract[extract$Experiment=="A",]
Q_F <- extract[extract$Experiment=="F",]
Q_AF <- extract[extract$Experiment=="AF",]
Q_EF <- extract[extract$Experiment=="EF",]
Q_AE <- extract[extract$Experiment=="AE",]

###venn subsets###
vA <- Q_A[,"UP_prot_ID"]
vF <- Q_F[,"UP_prot_ID"]
vAF <- Q_AF[,"UP_prot_ID"]
vAE <- Q_AE[,"UP_prot_ID"]
vEF <- Q_EF[,"UP_prot_ID"]
```

***

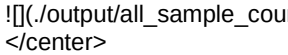
General considerations

Count across samples

```{r}
counts <- data.frame(table(extract$Prot_ID))
colnames(counts) <- c("Entry", "Count")
nrow(extract)
str(counts)
```

```{r}
tiff("./output/all_sample_count_prot_ID.tiff", height=300, width=300)
a = ggplot(counts, aes(Count)) + geom_histogram(binwidth=1, col="black", fill="blue", alpha=0.5)
a = a + geom_hline(yintercept=c(10,50,100,200,300,400), col="red", linetype="dashed")
a = a + theme_bw() + themeTIFF + xlab("Protein ID") + ylab("Count")
plot(a)
dev.off()
```

The total count of protein IDs across all the experiment types appears below.



Overrepresented proteins across samples

```{r}
###find proteins across all experiments###
##list these proteins for 4/5 occurrences###
overlap <- rbind(counts[counts$Count==4,],counts[counts$Count==5,])
overlap_match <- data.frame()
for(i in overlap$Entry){
  data <- extract[extract$Prot_ID==i,]
  data <- cbind(data, rep(i, nrow(data)))
  overlap_match <- rbind(data, overlap_match)
}
ID_count <- data.frame(table(overlap_match$Prot_ID))
ID_count <- ID_count[ID_count$Freq>=1,]
ID_count <- ID_count[ order(-ID_count$Freq), ]
colnames(ID_count) <- c("Protein_ID", "Count")
```

There were `r nrow(ID_count)` proteins that were identified across 4 or more experiment types (N = 5 total).

```{r}
IDs <- ID_count[ID_count$Count==5,]
IDmatch <- data.frame()

for(i in IDs$Protein_ID){
  data <- dataUP[dataUP$UP_prot_ID==i,]
  data$Protein_ID <- i
  IDmatch <- rbind(data, IDmatch)
}

COUNT5 <- data.frame(IDmatch$UP_prot_ID, IDmatch$Protein_ID, IDmatch$UP_prot_name)
colnames(COUNT5) <- c("UP_prot_ID", "Protein_ID", "UP_prot_name")
COUNT5

```

Count (independent AF searches)

```{r}
path <- "./input/IAPP_independent_searches/"

###function to extract proteinGroup files from independent Andromeda searches###
GETindependent <- function(x){
#see https://stackoverflow.com/questions/4876813/using-r-to-list-all-files-with-a-specified-extension
  files_IND <- list.files(path = x, pattern = "proteinGroups.txt", recursive=TRUE, include.dirs=TRUE,
ignore.case=TRUE)
  complete_group <- data.frame()
  for (i in files_IND){
#see https://stackoverflow.com/questions/8996134/extract-vectors-from-strsplit-list-without-using-a-loop
    Experiment <- rapply(strsplit(i, "/"), function(a) a[1])
    prot_group <- read.csv(paste(x, "/", i, sep=""), sep="\t", header=TRUE)
    prot_group$Experiment <- Experiment
    complete_group <- rbind(prot_group, complete_group)
  }
  return(complete_group)
}

###modify data frame###
protgroups_IND <- GETindependent(path)
protgroups_IND$Prot_ID <- rapply(strsplit(as.character(protgroups_IND$Protein.IDs),";"), function(a) a[1])

###check experiments included###
table(protgroups_IND$Experiment)

###subset based on experiment###
AF1 <- protgroups_IND[protgroups_IND$Experiment == "AF_1",]
```

```
AF2 <- protgroups_IND[protgroups_IND$Experiment == "AF_2",]
AF3 <- protgroups_IND[protgroups_IND$Experiment == "AF_3",]
AF4 <- protgroups_IND[protgroups_IND$Experiment == "AF_4",]
AF5 <- protgroups_IND[protgroups_IND$Experiment == "AF_5",]
AF6 <- protgroups_IND[protgroups_IND$Experiment == "AF_6",]

F1 <- protgroups_IND[protgroups_IND$Experiment == "F_1_156",]
F2 <- protgroups_IND[protgroups_IND$Experiment == "F_1_164",]
F3 <- protgroups_IND[protgroups_IND$Experiment == "F_1_170",]
F4 <- protgroups_IND[protgroups_IND$Experiment == "F_2_156",]
F5 <- protgroups_IND[protgroups_IND$Experiment == "F_2_170",]
F6 <- protgroups_IND[protgroups_IND$Experiment == "F_3_180",]
F7 <- protgroups_IND[protgroups_IND$Experiment == "F_4_180",]
F8 <- protgroups_IND[protgroups_IND$Experiment == "F_5_180",]
F9 <- protgroups_IND[protgroups_IND$Experiment == "F_6_180",]

###AF venn subsets###
vAF1 <- AF1[,"Prot_ID"]
vAF2 <- AF2[,"Prot_ID"]
vAF3 <- AF3[,"Prot_ID"]
vAF4 <- AF4[,"Prot_ID"]
vAF5 <- AF5[,"Prot_ID"]
vAF6 <- AF6[,"Prot_ID"]

###F venn subsets###
vF1 <- F1[,"Prot_ID"]
vF2 <- F2[,"Prot_ID"]
vF3 <- F3[,"Prot_ID"]
vF4 <- F4[,"Prot_ID"]
vF5 <- F5[,"Prot_ID"]
vF6 <- F6[,"Prot_ID"]
vF7 <- F7[,"Prot_ID"]
vF8 <- F8[,"Prot_ID"]
vF9 <- F9[,"Prot_ID"]

#see https://stackoverflow.com/questions/25019794/venn-diagram-with-item-labels
venn <- venn.diagram(list(AF1 = vAF1, AF2 = vAF2, AF3 = vAF3, AF4 = vAF4, AF5 = vAF5),
                     fill = c("red", "blue", "green", "yellow", "purple"),
                     alpha = 0.4, lwd = 1, cex = 1, cat.cex = 1, filename = NULL)

tiff("./output/venn_AF_independent.tiff", height=300, width=300)
grid.newpage()
grid.draw(venn)
dev.off()

venn <- venn.diagram(list(AF2 = vAF2, AF3 = vAF3, AF4 = vAF4, AF5 = vAF5, AF6= vAF6),
                     fill = c("blue", "green", "yellow", "purple", "brown"),
                     alpha = 0.4, lwd = 1, cex = 1, cat.cex = 1, filename = NULL)

tiff("./output/venn_AF_independent_2.tiff", height=300, width=300)
grid.newpage()
grid.draw(venn)
dev.off()

venn <- venn.diagram(list(F1 = vF1, F2 = vF2, F3 = vF3, F4 = vF4, F5 = vF5),
                     fill = c("blue", "green", "yellow", "purple", "orange"),
                     alpha = 0.4, lwd = 1, cex = 1, cat.cex = 1, filename = NULL)

tiff("./output/venn_F_independent.tiff", height=300, width=300)
grid.newpage()
grid.draw(venn)
dev.off()

venn <- venn.diagram(list(F3 = vF3, F4 = vF4, F5 = vF5, F6 = vF6, F7 = vF7),
                     fill = c("yellow", "purple", "orange", "brown", "pink"),
                     alpha = 0.4, lwd = 1, cex = 1, cat.cex = 1, filename = NULL)
```

```
tiff("./output/venn_F_independent_2.tiff", height=300, width=300)
grid.newpage()
grid.draw(venn)
dev.off()

venn <- venn.diagram(list(F5 = vF5, F6 = vF6, F7 = vF7, F8 = vF8, F9 = vF9),
                     fill = c("orange", "brown", "pink", "gray", "darkblue"),
                     alpha = 0.4, lwd = 1, cex = 1, cat.cex = 1, filename = NULL)

tiff("./output/venn_F_independent_3.tiff", height=300, width=300)
grid.newpage()
grid.draw(venn)
dev.off()
```

Global faceted analysis of variables

Unique peptides vs peptides
```{r}
#see http://ggplot2.tidyverse.org/reference/facet_grid.html
b <- ggplot(extract)
b <- b + geom_point(aes(Unique.peptides, Peptides), alpha=0.7, pch=21, fill="gray")
b <- b + facet_grid(. ~ Experiment)
b <- b + theme_bw() + theme(axis.text.x = element_text(angle=90, vjust=0.5))
plot(b)
```

GRAVY vs pI
```{r}
c <- ggplot(extract)
c <- c + geom_point(aes(GRAVY, pI), alpha=0.7, pch=21, fill="gray")
c <- c + facet_grid(. ~ Experiment)
c <- c + theme_bw()
plot(c)
```

GRAVY vs MW
```{r}
d <- ggplot(extract)
d <- d + geom_point(aes(GRAVY, MW), alpha=0.7, pch=21, fill="gray")
d <- d + facet_grid(. ~ Experiment)
d <- d + theme_bw() + scale_y_continuous(limits=c(0,1E+6))
plot(d)
```

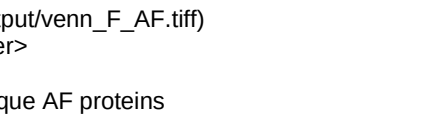
Question 1 - CC coronae

Overlap of AF/F protein IDs

Comparing FBS only (F), with IAPP amyloids exposed to FBS (AF). These samples were prepared with 1 MDa molecular weight cut-off (MWCO) spin columns (CC method).

The identification overlap was as follows:
```{r}
#see https://stackoverflow.com/questions/25019794/venn-diagram-with-item-labels
venn <- venn.diagram(list(F = vF, AF = vAF), fill = c("red", "blue"),
                     alpha = 0.4, lwd = 1, cex = 1.5, cat.cex = 1.5, filename = NULL)
```

```
tiff("./output/venn_F_AF.tiff", height=300, width=300)
grid.newpage()
grid.draw(venn)
dev.off()
```



Unique AF proteins

Here, plots are created that summarise the physico-chemical properties of proteins unique to the AF samples, highlighting some potentially interesting amyloid binding proteins that could be investigated in future.

Extracting unique entries from combined AF/F data:
```{r}
#see https://stackoverflow.com/questions/16905425/find-duplicate-values-in-r
AF_F_ID_count <- data.frame(table(Q_AF_F$Prot_ID))
AF_F_ID_count <- AF_F_ID_count[AF_F_ID_count$Freq==1,]

###extract unique entries for AF and F###
AF_F_unique <- data.frame()
for(i in AF_F_ID_count$Var1){
  data <- Q_AF_F[Q_AF_F$Prot_ID==i,]
  data <- cbind(data, i)
  AF_F_unique <- rbind(data, AF_F_unique)
}
###check dataframe###
#head(AF_F_unique)
```

Confirming Venn diagram counts:
```{r}
AF_F_unique_count <- data.frame(table(AF_F_unique$Experiment))
colnames(AF_F_unique_count) <- c("Prot_ID", "Count")
AF_F_unique_count
```

Extract unique AF proteins and export these for STRING network analysis:
```{r}
###create subset dataframe###
AF <- AF_F_unique[AF_F_unique$Experiment=="AF",]
F <- AF_F_unique[AF_F_unique$Experiment=="F",]

#see http://r.789695.n4.nabble.com/write-text-file-as-output-without-quotes-td888020.html
write.table(AF$Prot_ID, "./output/exp_AF_protlist.txt", sep="\t", quote=FALSE, row.names = FALSE, col.names = FALSE)
```

The top 12 unique AF proteins, based on unique peptide count were:
```{r}
#see https://stackoverflow.com/questions/1296646/how-to-sort-a-dataframe-by-columns
uniqueAF_sort <- AF[ order(-AF$Unique.peptides), ]
uniqueAF_sort <- data.frame(uniqueAF_sort$Prot_ID, uniqueAF_sort$UP_prot_name, uniqueAF_sort$Unique.peptides)
colnames(uniqueAF_sort) <- c("Protein ID", "Protein name", "Unique peptides")
uniqueAF_sort[1:12,]
```

Unique AF/F amino acid composition

Concept inspired by a table in the following reference -> DOI:10.1039/C7EN00466D (see http://pubs.rsc.org/en/content/articlehtml/2017/en/c7en00466d)

```{r}
###experimental###
```

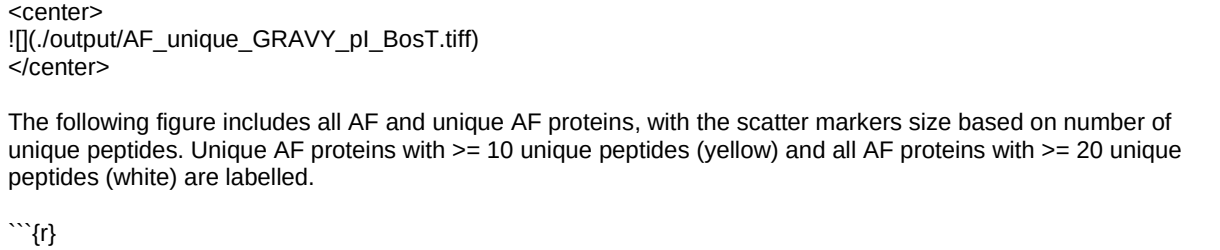
```
propAF <- data.frame()
for (i in AF_F_unique$UP_sequence){
  seq_vec <- unlist(strsplit(as.character(i), ""))
  info <- AAstat(seq_vec, plot=FALSE)
  all <- AF_F_unique[AF_F_unique$UP_sequence==i,]
  summary <- cbind(all, info$Prop$Aliphatic, info$Prop$Aromatic, info$Prop$Non.polar,
                   info$Prop$Polar, info$Prop$Charged, info$Prop$Basic, info$Prop$Acidic)
  propAF <- rbind(summary, propAF)
}

###uniprot###
propUP <- data.frame()
for (i in dataUP$UP_sequence){
  seq_vec <- unlist(strsplit(as.character(i), ""))
  info <- AAstat(seq_vec, plot=FALSE)
  all <- dataUP[dataUP$UP_sequence==i,]
  summary <- cbind(all, info$Prop$Aliphatic, info$Prop$Aromatic, info$Prop$Non.polar,
                   info$Prop$Polar, info$Prop$Charged, info$Prop$Basic, info$Prop$Acidic)
  propUP <- rbind(summary, propUP)
}

###subset based on specific columns###
propAF <- propAF[,c(1,2,3,40,41,43,44,45,46,47,48,49)]
###check colnames###
colnames(propAF)
colnames(propUP)

###update column names###
colnames(propAF) <- c("UP_prot_ID", "UP_prot_name", "UP_gene", "Experiment", "Prot_ID", "Aliphatic",
                      "Aromatic", "NonPolar", "Polar", "Charged", "Basic", "Acidic")

colnames(propUP) <- c("UP_prot_ID", "UP_prot_name", "UP_gene", "UP_sequence", "pI", "MW", "GRAVY",
"Aliphatic",
                      "Aromatic", "NonPolar", "Polar", "Charged", "Basic", "Acidic")

###plots###
a <- ggplot(propUP) + geom_point(aes(NonPolar, Acidic), col="gray", alpha=0.4)
a <- a + geom_point(data=propAF[propAF$Experiment=="F",], aes(NonPolar, Acidic), fill = "blue", pch=21)
a <- a + geom_point(data=propAF[propAF$Experiment=="AF",], aes(NonPolar, Acidic), fill = "yellow", pch=21)
a <- a + theme_bw() + coord_fixed()
plot(a)
```

Unique AF proteins (GRAVY/pI + proteome)

Here, the Bos taurus proteome (gray) is overlaid with all unique FBS only proteins (F, blue), as well as those proteins unique to amyloid + FBS (AF, yellow). Labels are gene names, outside the box indicated on the plot. The top panel labels unique AF proteins outside the boundaries indicated by the highlighted blue box. The bottom panel labels unique AF proteins (GRAVY <=-1 and GRAVY >=0 and pI <= 4.5 and pI >= 7).

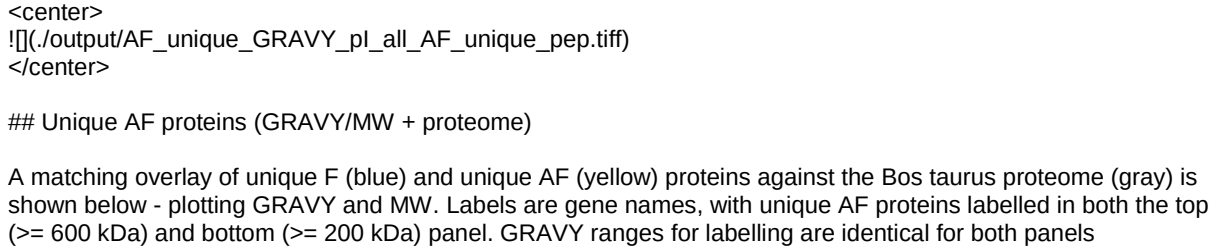
```{r}
###plot###
tiff("./output/AF_unique_GRAVY_pI_BosT.tiff", width=2500, height=5000, res=300)
#themes, see #http://ggplot2.tidyverse.org/reference/theme.html
#geom_rect, see - https://stackoverflow.com/questions/4733182/how-to-highlight-time-ranges-on-a-plot
a <- ggplot(dataUP)
a <- a + geom_point(data=dataUP, aes(GRAVY, pI), col="gray", alpha=0.1)
a <- a + geom_point(data=F, aes(GRAVY, pI), pch=21, size=5, fill="blue")
a <- a + geom_point(data=AF, aes(GRAVY, pI), pch=21, size=3, fill="yellow")
a <- a + theme_bw() + themeTIFF + xlab("GRAVY") + ylab("pI")
b <- a + geom_rect(data=data.frame(xmin=-1.5, xmax=0.5, ymin=4, ymax=10),
                   aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
                   fill="lightblue", col="black", alpha=0.4,
                   inherit.aes = FALSE)
b <- b + geom_label_repel(data=AF[AF$GRAVY<=-1.5 | AF$GRAVY>=0.5 | AF$pI<=4 | AF$pI>=10,],
                          aes(GRAVY, pI, label=UP_gene), size=4)
```

```
c <- a + scale_y_continuous(limits=c(4,10)) + scale_x_continuous(limits=c(-1.5,0.5))
c <- c + geom_rect(data=data.frame(xmin=-1, xmax=0, ymin=4.5, ymax=7),
                   aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
                   fill="lightblue", col="black", alpha=0.4,
                   inherit.aes = FALSE)
c <- c + geom_label_repel(data=AF[AF$GRAVY<=-1 | AF$GRAVY>=0 | AF$pI<=4.5 | AF$pI>=7,],
                          aes(GRAVY, pI, label=UP_gene), size=4)
c <- c + guides(fill=FALSE)
grid.arrange(b, c, nrow=2)
dev.off()
```



The following figure includes all AF and unique AF proteins, with the scatter markers size based on number of unique peptides. Unique AF proteins with >= 10 unique peptides (yellow) and all AF proteins with >= 20 unique peptides (white) are labelled.

```{r}
###plot###
tiff("./output/AF_unique_GRAVY_pI_all_AF_unique_pep.tiff", width=700, height=700)
d <- ggplot(Q_AF) + geom_point(aes(GRAVY, pI, size=Unique.peptides), col="blue")
d <- d + geom_point(data=AF, aes(GRAVY, pI, size=Unique.peptides), col="yellow")
d <- d + scale_x_continuous(limits=c(-1,0)) + scale_y_continuous(limits=c(4,7.5))
d <- d + geom_label_repel(data=Q_AF[Q_AF$Unique.peptides>=20,], aes(GRAVY, pI, label=UP_gene), size=3)
d <- d + geom_label_repel(data=AF[AF$Unique.peptides>=10,], aes(GRAVY, pI, label=UP_gene), fill="yellow",
size=3)
d <- d + theme_bw() + themeTIFF + xlab("GRAVY") + ylab("pI")
d <- d + theme(panel.background = element_rect(fill="gray"), panel.grid.minor = element_line(colour="gray"),
               panel.grid.major = element_line(colour="gray"))
plot(d)
dev.off()
```



Unique AF proteins (GRAVY/MW + proteome)

A matching overlay of unique F (blue) and unique AF (yellow) proteins against the Bos taurus proteome (gray) is shown below - plotting GRAVY and MW. Labels are gene names, with unique AF proteins labelled in both the top (>= 600 kDa) and bottom (>= 200 kDa) panel. GRAVY ranges for labelling are identical for both panels (GRAVY >= 0.5 & <= -1.5)

```{r}
###plot###
tiff("./output/AF_unique_GRAVY_MW_BosT.tiff", width=2500, height=5000, res=300)
a <- ggplot(dataUP)
a <- a + geom_point(data=dataUP, aes(GRAVY, MW), col="gray", alpha=0.1)
a <- a + geom_point(data=F, aes(GRAVY, MW), pch=21, size=5, fill="blue")
a <- a + geom_point(data=AF, aes(GRAVY, MW), pch=21, size=3, fill="yellow", col="blue")
a <- a + theme_bw() + themeTIFF + xlab("GRAVY") + ylab("MW")
b <- a + geom_rect(data=data.frame(xmin=-1.5, xmax=0.5, ymin=0, ymax=6E+5),
                   aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
                   fill="lightblue", col="black", alpha=0.4, inherit.aes = FALSE)
b <- b + geom_label_repel(data=AF[AF$GRAVY<=-1.5 | AF$GRAVY>=0.5 | AF$MW>=6E+5,],
                          aes(GRAVY, MW, label=UP_gene), size=4)
c <- a + scale_y_continuous(limits=c(0,6E+5)) + scale_x_continuous(limits=c(-1.5,0.5))
c <- c + geom_label_repel(data=AF[AF$GRAVY<=-1.5 | AF$GRAVY>=0.5 | AF$MW>=2E+5,],
                          aes(GRAVY, MW, label=UP_gene), size=4)
c <- c + guides(fill=FALSE)
grid.arrange(b, c, nrow=2)
dev.off()
```


Question 2 - QCM coronae

Overlap of AE/EF protein IDs

The identification overlap was as follows:
```{r}
#see https://stackoverflow.com/questions/25019794/venn-diagram-with-item-labels
venn <- venn.diagram(list(AE = vAE, EF = vEF), fill = c("yellow", "blue"),
                     alpha = 0.4, lwd = 1, cex = 1.5, cat.cex = 1.5, filename = NULL)

tiff("./output/venn_AE_EF.tiff", width=2500, height=2500, res=300)
grid.newpage()
grid.draw(venn)
dev.off()
```


Overlay of unique AE/EF proteins
```{r}
#see https://stackoverflow.com/questions/16905425/find-duplicate-values-in-r
AE_EF_ID_count <- data.frame(table(Q_AE_EF$Prot_ID))
AE_EF_ID_count <- AE_EF_ID_count[AE_EF_ID_count$Freq==1,]

AE_EF_unique <- data.frame()
for(i in AE_EF_ID_count$Var1){
  data <- Q_AE_EF[Q_AE_EF$Prot_ID==i,]
  data <- cbind(data, i)
  AE_EF_unique <- rbind(data, AE_EF_unique)
}

AE_EF <- data.frame(table(AE_EF_unique$Experiment))
AE_EF <- AE_EF[AE_EF$Freq>0,]
colnames(AE_EF) <- c("Prot_ID", "Count")
Q2_AE <- AE_EF_unique[AE_EF_unique$Experiment=="AE",]
Q2_EF <- AE_EF_unique[AE_EF_unique$Experiment=="EF",]
```

There were a total of `r nrow(Q_EF)` and `r nrow(Q_AE)` IDs for experiments EF and AE, respectively. Of these there were `r nrow(Q2_EF)` and `r nrow(Q2_AE)` unique protein identifications for EF and AE, respectively.

The top 12 unique AE proteins, based on unique peptide count were:
```{r}
uniqueAE_sort <- Q2_AE[ order(-Q2_AE$Unique.peptides), ]
uniqueAE_sort <- data.frame(uniqueAE_sort$Prot_ID, uniqueAE_sort$UP_prot_name,
uniqueAE_sort$Unique.peptides)
colnames(uniqueAE_sort) <- c("Protein ID", "Protein name", "Unique peptides")
uniqueAE_sort[1:12,]
```

Here, the protein list for unique AE proteins was exported for STRING DB analysis.
```{r export_AE}
#see http://r.789695.n4.nabble.com/write-text-file-as-output-without-quotes-td888020.html
write.table(Q2_AE$Prot_ID, "./output/exp_AE_protlist.txt", sep="\t", quote=FALSE, row.names = FALSE,
col.names = FALSE)
```

The unique EF proteins were:
```{r}
```

```
uniqueEF_sort <- Q2_EF[ order(-Q2_EF$Unique.peptides), ]
uniqueEF_sort <- data.frame(uniqueEF_sort$Prot_ID, uniqueEF_sort$UP_prot_name,
uniqueEF_sort$Unique.peptides)
colnames(uniqueEF_sort) <- c("Protein ID", "Protein name", "Unique peptides")
uniqueEF_sort
```

Unique AE/EF amino acid composition

Concept inspired by a table in the following reference -> DOI:10.1039/C7EN00466D (see http://pubs.rsc.org/en/content/articlehtml/2017/en/c7en00466d)

```{r}
propAE <- data.frame()
for (i in AE_EF_unique$UP_sequence){
  seq_vec <- unlist(strsplit(as.character(i), ""))
  info <- AAstat(seq_vec, plot=FALSE)
  all <- AE_EF_unique[AE_EF_unique$UP_sequence==i,]
  summary <- cbind(all, info$Prop$Aliphatic, info$Prop$Aromatic, info$Prop$Non.polar,
                   info$Prop$Polar, info$Prop$Charged, info$Prop$Basic, info$Prop$Acidic)
  propAE <- rbind(summary, propAE)
}

###subset and rename columns###
propAE <- propAE[,c(1,2,3,40,41,43,44,45,46,47,48,49)]
###check column names###
colnames(propAE)
###update column names###
colnames(propAE) <- c("UP_prot_ID", "UP_prot_name", "UP_gene", "Experiment", "Prot_ID", "Aliphatic",
                      "Aromatic", "NonPolar", "Polar", "Charged", "Basic", "Acidic")

###plots###
a <- ggplot(propUP) + geom_point(aes(NonPolar, Acidic), col="gray", alpha=0.4)
a <- a + geom_point(data=propAE[propAE$Experiment=="AE",], aes(NonPolar, Acidic), fill = "yellow", pch=21)
a <- a + geom_point(data=propAE[propAE$Experiment=="EF",], aes(NonPolar, Acidic), fill = "blue", pch=21)
a <- a + theme_bw() + coord_fixed()
plot(a)

```

Unique AE/EF proteins (GRAVY/pI/MW + proteome)
Here, the Bos taurus proteome (gray) is overlaid with unique AE (yellow) and unique EF (blue) proteins - plotting GRAVY/pI and GRAVY/MW.
```{r}
tiff("./output/AE_EF_unique_GRAVY_pI_BosT.tiff", width=2500, height=5000, res=300)
#themes, see #http://ggplot2.tidyverse.org/reference/theme.html
d <- ggplot(dataUP)
d <- d + geom_point(aes(GRAVY, pI), size=1, col="gray", alpha=0.5)
d <- d + geom_point(data=Q2_EF, aes(GRAVY, pI), pch=21, fill="blue", size=5)
d <- d + geom_point(data=Q2_AE, aes(GRAVY, pI), pch=21, fill="yellow", size=3)
d <- d + xlab("GRAVY") + ylab("pI") + theme_bw() + themeTIFF
#geom_rect, see - https://stackoverflow.com/questions/4733182/how-to-highlight-time-ranges-on-a-plot
e <- d + geom_rect(data=data.frame(xmin=-1.5, xmax=0.5, ymin=4, ymax=10),
                   aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
                   fill="lightblue", col="black", alpha=0.4, inherit.aes = FALSE)
e <- e + geom_label_repel(data=Q2_AE[Q2_AE$pI>=10 | Q2_AE$pI<=4 | Q2_AE$GRAVY>=0.5 |
Q2_AE$GRAVY<=-1.5,],
                          aes(GRAVY, pI, label=UP_gene), size=4)
f <- d + scale_x_continuous(limits=c(-1.5,1)) + scale_y_continuous(limits=c(4,10))
f <- f + geom_rect(data=data.frame(xmin=-1, xmax=0, ymin=4.5, ymax=8.5),
                   aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
                   fill="lightblue", col="black", alpha=0.4, inherit.aes = FALSE)
f <- f + geom_label_repel(data=Q2_AE[Q2_AE$pI>=8.5 | Q2_AE$pI<=4.5 | Q2_AE$GRAVY>=0 |
Q2_AE$GRAVY<=-1,],
                          aes(GRAVY, pI, label=UP_gene), size=4)
grid.arrange(e, f, nrow=2)
dev.off()
```

```

<center>
![](./output/AE_EF_unique_GRAVY_pI_BosT.tiff)
</center>

```{r}
tiff("./output/AE_EF_unique_GRAVY_MW_BosT.tiff", width=2500, height=5000, res=300)
#themes, see #http://ggplot2.tidyverse.org/reference/theme.html
d <- ggplot(dataUP)
d <- d + geom_point(aes(GRAVY, MW), size=1, col="gray", alpha=0.5)
d <- d + geom_point(data=Q2_EF, aes(GRAVY, MW), pch=21, fill="blue", size=5)
d <- d + geom_point(data=Q2_AE, aes(GRAVY, MW), pch=21, fill="yellow", size=3)
d <- d + xlab("GRAVY") + ylab("MW") + theme_bw() + themeTIFF
#geom_rect, see - https://stackoverflow.com/questions/4733182/how-to-highlight-time-ranges-on-a-plot
e <- d + geom_rect(data=data.frame(xmin=-1.5, xmax=0.5, ymin=0, ymax=600000),
                   aes(xmin=xmin, xmax=xmax, ymin=ymin, ymax=ymax),
                   fill="lightblue", col="black", alpha=0.4, inherit.aes = FALSE)
e <- e + geom_label_repel(data=Q2_AE[Q2_AE$MW>=6E+5 | Q2_AE$GRAVY>=0.5 | Q2_AE$GRAVY<=-1.5,],
                          aes(GRAVY, MW, label=UP_gene), size=4)
f <- d + scale_y_continuous(limits=c(0,600000))
f <- f + geom_label_repel(data=Q2_AE[Q2_AE$MW>=3E+5 | Q2_AE$GRAVY>=0.5 | Q2_AE$GRAVY<=-1.5,],
                          aes(GRAVY, MW, label=UP_gene), size=4)
grid.arrange(e, f, nrow=2)
dev.off()
```

<center>

</center>

***

Question 3 - CC vs QCM

Overlap of AF/AE protein IDs
The next comparison of interest was to consider the difference, if any, between the amyloids + FBS prepared by either the CC method (AF) or the QCM method (AE).

The identification overlap for these experiments was as follows:
```{r}
#see https://stackoverflow.com/questions/25019794/venn-diagram-with-item-labels
venn <- venn.diagram(list(AF = vAF, AE = vAE), fill = c("red", "blue"),
                     alpha = 0.4, lwd = 1, cex = 2.5, cat.cex = 2.5, filename = NULL)

tiff("./output/venn_AF_AE.tiff", width=2500, height=2500, res=300)
grid.newpage()
grid.draw(venn)
dev.off()
```

<center>

</center>

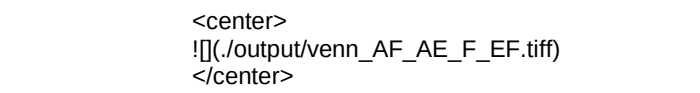
There were a total of `r nrow(Q_AF)` and `r nrow(Q_AE)` IDs for experiments AF and AE, respectively.

If we consider the Venn diagram showing the four experiments F/AF/AE/EF, we can identify those proteins unique to AF/AE.

```{r}
#see https://stackoverflow.com/questions/25019794/venn-diagram-with-item-labels
venn <- venn.diagram(list(AF = vAF, AE = vAE, F=vF, EF=vEF), fill = c("red", "blue", "green", "yellow"),
                     alpha = 0.4, lwd = 1, cex = 1.5, cat.cex = 1.5, filename = NULL)

tiff("./output/venn_AF_AE_F_EF.tiff", height=300, width=300)
grid.newpage()
```
```

```
grid.draw(venn)
dev.off()
```



***

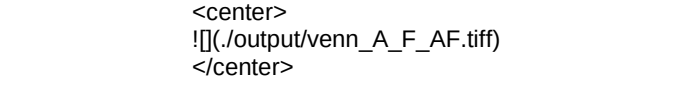
Question 4 - Control nLC-MS/MS
A

Amyloid only control for CC method.

Sample A nLC-MS/MS analyses identified a total of `r nrow(Q_A)` proteins.

These proteins were:
```{r}
Q_A_sort <- Q_A[ order(-Q_A$Unique.peptides), ]
A_list <- data.frame(Q_A_sort$Prot_ID, Q_A_sort$UP_prot_name, Q_A_sort$Unique.peptides)
colnames(A_list) <- c("Protein ID", "Protein name", "Unique peptides")
A_list[1:12,]
```

The identification overlap for A/F/AF was:
```{r}
#see https://stackoverflow.com/questions/25019794/venn-diagra♣m-with-item-labels
venn <- venn.diagram(list(A = vA, F = vF, AF=vAF), fill = c("green", "blue", "yellow"),
                     alpha = 0.4, lwd = 1, cex = 2.5, cat.cex = 2.5, filename = NULL)

tiff("./output/venn_A_F_AF.tiff", width=2500, height=2500, res=300)
grid.newpage()
grid.draw(venn)
dev.off()

```



A (using human .fasta file)

```{r}
A_human <- read.csv("./input/Ahuman/combined/txt/proteinGroups.txt", sep="\t", header=TRUE)
A_human$Prot_ID <- rapply(strsplit(as.character(A_human$Protein.IDs),";"), function(a) a[1])
A_human$Prot_name <- rapply(strsplit(as.character(A_human$Protein.names),";"), function(a) a[1])
```

Sample A nLC-MS/MS analyses using a human fasta file for the Andromeda search identified a total of `r nrow(A_human)` proteins.

The top 12 proteins in this list, based on unique peptides counts, were:
```{r}
A_human_sort <- A_human[ order(-A_human$Unique.peptides), ]
A_human_sort <- data.frame(A_human_sort$Prot_ID, A_human_sort$Prot_name,
A_human_sort$Unique.peptides)
colnames(A_human_sort) <- c("Protein ID", "Protein name", "Unique peptides")
A_human_sort[1:12,]
```

***

Extract data consistency check

Random subset of 30 rows from extract dataframe checked to make sure that IDs, names and sequences align correctly.

```{r}
# see https://stackoverflow.com/questions/8273313/sample-random-rows-in-dataframe
CHECK <- extract[sample(nrow(extract), 30), ]
write.table(CHECK, "./output/extract_consistency_check.txt", sep="\t")
```

***

Check of MaxQuant parameters
This section checks the consistency of the parameters used for MaxQuant.

```{r}
path <- "./input/IAPP"
GETparameters <- function(x){
  #see https://stackoverflow.com/questions/4876813/using-r-to-list-all-files-with-a-specified-extension
  parameter_files <- list.files(path = x, pattern = "parameters.txt", recursive=TRUE, include.dirs=TRUE,
ignore.case = TRUE)
  complete_parameters <- data.frame()
  complete_parameters <- read.csv(paste(x, "/", parameter_files[[1]], sep=""), sep="\t", header=TRUE)$Parameter
  for (i in parameter_files){
    #see https://stackoverflow.com/questions/8996134/extract-vectors-from-strsplit-list-without-using-a-loop
    Experiment <- rapply(strsplit(i, "/"), function(a) a[1])
    parameters <- read.csv(paste(x, "/", i, sep=""), sep="\t", header=TRUE)
    colnames(parameters) <- c("Parameter", Experiment)
    # see http://www.r-tutor.com/r-introduction/data-frame/data-frame-column-slice
    complete_parameters <- cbind(complete_parameters, parameters[2])
  }
  return(complete_parameters)
}

parameters <- GETparameters(path)

#see https://stackoverflow.com/questions/28628384/count-number-of-unique-values-per-row
apply(parameters[,2:6], 1, function(x)length(unique(x)))
```

Row four contains multiple unique entries. This row contains reports the following variable:

```{r}
parameters[4,1]
```

***

Chromatogram plotting script

Package xcms license is GPL (>= 2) + a file LICENSE available @
https://bioconductor.org/packages/release/bioc/licenses/xcms/LICENSE.

```{r}
#source("https://bioconductor.org/biocLite.R")
#biocLite("xcms")

#########################
###libraries required###
#########################
#library(xcms)
#library(ggplot2)
#library(gridExtra)
#library(stringr)
#library(plyr)

###############
###citations###
```

```
###############
#citation("xcms")
#citation("ggplot2")
#citation("gridExtra")
#citation("stringr")

###########
###notes###
###########
#see https://bioconductor.org/packages/3.7/bioc/vignettes/xcms/inst/doc/new_functionality.html
#see https://bioconductor.org/packages/release/bioc/vignettes/xcms/inst/doc/xcms.html
#see https://www.rdocumentation.org/packages/xcms/versions/1.48.0/topics/xcmsRaw
#these vignettes, documentation and examples were used in writing of the code provided below.

#########################################################################
###############set data directory and file_type#####################
###mzR can load many file types - this code was written for mzML###
#########################################################################
#path <- "./Converted_RAW_files/"
#file_type <- ".mzML"

#####################################################
###################lcms_vis#########################
###FUNCTION TO CREATE TIC CHROMATOGRAM OUTPUTS###
#####################################################

#lcms_vis <- function(x){
#  #see https://stackoverflow.com/questions/4876813/using-r-to-list-all-files-with-a-specified-extension
#  files <- list.files(path = x, pattern= paste("\\", file_type, sep=""))
#  complete_TIC_data <- data.frame()
#  for (i in files){
#    #see https://stackoverflow.com/questions/8996134/extract-vectors-from-strsplit-list-without-using-a-loop
#    id <- rapply(strsplit(i, "_"), function(a) a[6])
#    ID <- rapply(strsplit(id, ".m"), function(a) a[1])
#    #see https://stackoverflow.com/questions/18115550/how-to-combine-two-or-more-columns-in-a-dataframe-
into-a-new-column-with-a-new-n
#    uniqueID <- paste(rapply(strsplit(i,"_"), function(a) a[4]),"_", ID, sep="")
#    #see https://stackoverflow.com/questions/9756360/r-split-character-data-into-numbers-and-letters
#    replicate <- as.numeric(str_extract(ID, "[0-9]+"))
#    experiment <- (str_extract(ID, "[aA-zZ]+"))
#    #see https://www.rdocumentation.org/packages/xcms/versions/1.48.0/topics/xcmsRaw
#    data <- xcmsRaw(paste(path, i, sep=""), profstep = 1, profmethod = "bin", profparam = list(),
#                    includeMSn=FALSE, mslevel=NULL, scanrange=NULL)
#    tiff(file=paste("./chromatograms/", uniqueID, "_combined_chromatograms.tiff", sep=""), width=1500,
height=750)
#    #see https://www.statmethods.net/advgraphs/layout.html
#    par(mfrow=c(1,2))
#    plotChrom(data, base=TRUE)
#    plotChrom(data, base=FALSE)
#    dev.off()
#    TIC <- data.frame(uniqueID, ID, experiment, replicate, data@scantime, data@tic)
#    complete_TIC_data <- rbind(complete_TIC_data, TIC)
#  }
#  return(complete_TIC_data)
#}

#################################
###Process selected directory###
#################################
#DATA <- lcms_vis(path)
#write.table(DATA, "./complete_TIC_data.txt", sep="\t")

#####################################
###Create combined chromatograms###
#####################################

#for (i in unique(DATA$experiment)){
```

```
#  plotDATA <- subset(DATA, DATA$experiment==i)
#  pdf(paste("./chromatogram_overlays/chromatograms_",i,".pdf",sep=""), width=15, height=15)
#  #see http://ggplot2.tidyverse.org/reference/scale_brewer.html
#  a = ggplot(plotDATA, aes(data.scantime, data.tic)) + geom_line(alpha=0.5, aes(col=as.character(replicate)))
#  #see https://stackoverflow.com/questions/14622421/how-to-change-legend-title-in-ggplot
#  a = a + ggtitle(paste("LC-MS/MS chromatograms for ", i, " experiments", sep=""))
#  a = a + xlab("Scan time") + ylab("total ion count (TIC)") + labs(color="Experiment No.")
#  #see http://ggplot2.tidyverse.org/reference/theme.html
#  a = a + theme_bw() + theme(legend.text = element_text(size = 20), legend.title=element_text(size=25),
#                         axis.title=element_text(size=25), legend.key.size = unit(0.8, "cm"),
#                         axis.text.x = element_text(size=20), axis.text.y = element_text(size=20),
#                         plot.title = element_text(size=25))
#  plot(a)
#  dev.off()
#}

###session info###
#sessionInfo()
```

***

Attributions

General attributions

Description	Link
Rmarkdown basics	http://rmarkdown.rstudio.com/authoring_basics.html
Rmarkdown format	http://rmarkdown.rstudio.com/markdown_document_format.html
In-line R code	http://rmarkdown.rstudio.com/lesson-4.html
Degree symbol	https://stackoverflow.com/questions/26234119/how-can-i-write-a-degree-symbol-in-rmarkdown2-for-html-and-pdf-output
Tables	https://stackoverflow.com/questions/19997242/simple-manual-rmarkdown-tables-that-look-good-in-html-pdf-and-docx
Tables	https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf
Table dimensions	https://stackoverflow.com/questions/27219629/how-to-control-cell-width-in-pandoc-markdown-table
Rmarkdown themes	http://rmarkdown.rstudio.com/html_document_format.html
Rmarkdown image formatting	https://stackoverflow.com/questions/15625990/how-to-set-size-for-local-image-using-knitr-for-markdown
Image formatting	https://stackoverflow.com/questions/24677642/centering-image-and-text-in-r-markdown-for-a-pdf-report
css fonts	https://stackoverflow.com/questions/29274501/r-markdown-changing-font-size-and-font-type-in-html-output
css style	https://stackoverflow.com/questions/30446905/rmarkdown-font-size-and-header
css body/TOC modifications	https://rpubs.com/stevepowell99/floating-css
TOC	https://stackoverflow.com/questions/42546001/how-to-change-colors-and-attributes-of-table-of-contents-in-r-markdown-html-docu?rq=1
reshape	https://www.statmethods.net/management/reshape.html
ggrepel	https://cran.r-project.org/web/packages/ggrepel/vignettes/ggrepel.html
themes	http://ggplot2.tidyverse.org/reference/theme.html
geom_rect	https://stackoverflow.com/questions/4733182/how-to-highlight-time-ranges-on-a-plot
ggplot theme	https://rstudio-pubs-static.s3.amazonaws.com/3364_d1a578f521174152b46b19d0c83cbe7e.html
Stand alone error	https://github.com/rstudio/rmarkdown/issues/228

|Subsetting |http://adv-r.had.co.nz/Subsetting.html
|
|TIFF resolution |https://www.r-bloggers.com/high-resolution-figures-in-r/ & https://stat.ethz.ch/pipermail/r-help/2010-August/250893.html |

Additional attributions appear throughout this document. Citations, session information and package links appear below.

Session info
```{r sessioninfo}
sessionInfo()
```
Package links
Name	URL
gridExtra	https://CRAN.R-project.org/package=gridExtra
ggplot2	http://ggplot2.org
seqinr	https://cran.r-project.org/web/packages/seqinr/
alakazam	http://doi.org/10.1126/scitranslmed.3008879
sqldf	https://CRAN.R-project.org/package=sqldf
ggrepel	https://CRAN.R-project.org/package=ggrepel
VennDiagram	https://cran.r-project.org/web/packages/VennDiagram/index.html
stringr	https://cran.r-project.org/web/packages/stringr/index.html
reshape2	https://cran.r-project.org/web/packages/reshape2/
xcms	https://bioconductor.org/packages/release/bioc/html/xcms.html
Package citations
```{r citations}
citation("gridExtra")
citation("ggplot2")
citation("seqinr")
citation("alakazam")
citation("sqldf")
citation("ggrepel")
citation("VennDiagram")
citation("stringr")
citation("reshape2")
#citation("xcms")
```

References

1. R Core Team, R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria. [Online] **2014**, http://www.R-project.org/ (accessed Feb 19, 2018).
2. Charif, D.; J. R. Lobry, SeqinR 1.0-2: A Contributed Package to the R Project for Statistical Computing Devoted to Biological Sequences Retrieval and Analysis. In *Structural Approaches to Sequence Evolution: Molecules, Networks, Populations*; Bastolla, U., Porto, M., Roman, H. E., Vendruscolo, M., Eds.; Springer: Berlin, Heildelberg, 2007; pp 207-232.

3. Gupta, N. T.; Vander Heiden, J. A.; Uduman, M.; Gadala-Maria, D.; Yaari, G.; Kleinstein, S. H., Change-O: A Toolkit for Analyzing Large-scale B Cell Immunoglobulin Repertoire Sequencing Data. *Bioinformatics* **2015**, *31*, 3356-3358.
4. Baptiste, A. gridExtra: Miscellaneous Functions for "Grid" Graphics. [Online] **2017**, https://CRAN.R-project.org/package=gridExtra, R-project (accessed Feb 19, 2018).
5. Chen, H. VennDiagram: Generate High-Resolution Venn and Euler Plots. [Online] **2017**, https://CRAN.R-project.org/package=VennDiagram, R-project (accessed Feb 19, 2018).
6. Wickham, H. ggplot2: Elegant Graphics for Data Analysis. [Online] **2009**, http://ggplot2.org, Springer-Verlag New York (accessed Feb 19, 2018).
7. Szklarczyk, D.; Franceschini, A.; Wyder, S.; Forslund, K.; Heller, D.; Huerta-Cepas, J.; Simonovic, M.; Roth, A.; Santos, A.; Tsafou, K. P.; Kuhn, M.; Bork, P.; Jensen, L. J.; von Mering, C., STRING v10: Protein-Protein Interaction Networks, Integrated over the Tree of Life. *Nucleic Acids Res.* **2015**, *45*, D447-D452.
8. Szklarczyk, D.; Morris, J. H.; Cook, H.; Kuhn, M.; Wyder, S.; Simonovic, M.; Santos, A.; Doncheva, N. T.; Roth, A.; Bork, P.; Jensen, L. J.; von Mering, C., The STRING Database in 2017: Quality-Controlled Protein–Protein Association Networks, Made Broadly Accessible. *Nucleic Acids Res.* **2017,** *43*, D362-D368.
9. Zheng, X.; Baker, H.; Hancock, W. S.; Fawaz, F.; McCaman, M.; Pungor, E., Jr., Proteomic Analysis for the Assessment of Different Lots of Fetal Bovine Serum as a Raw Material for Cell Culture. Part IV. Application of Proteomics to the Manufacture of Biological Drugs. *Biotechnol. Prog.* **2006,** *22,* 1294-1300.