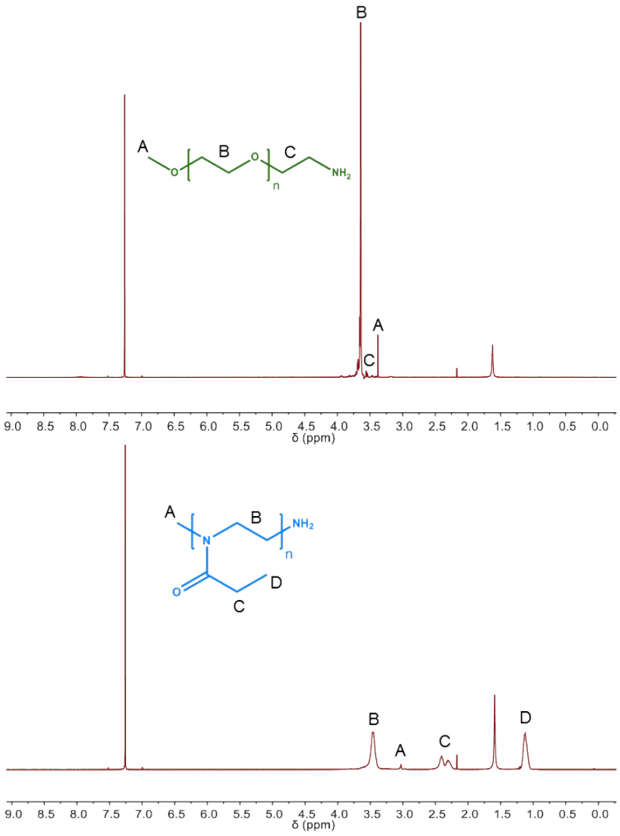
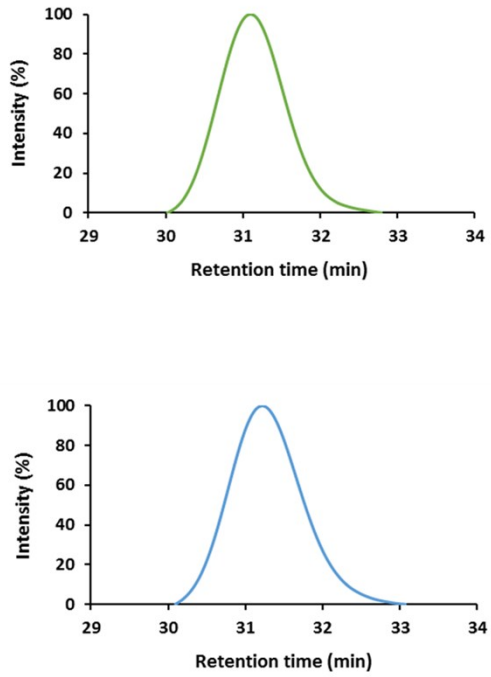
Electronic Supplementary Information

Human plasma proteome association and cytotoxicity of nano-graphene oxide grafted with stealth polyethylene glycol and poly(2-ethyl-2-oxazoline)

Miaoyi Wang,[†] Ove J. R. Gustafsson,[¶,≈] Ghizal Siddiqui,[‡,≈] Ibrahim Javed,[†] Hannah G. Kelly[ǁ,∏] Thomas Blin,[†] Hong Yin,[∫] Stephen J. Kent,[ǁ,∏,◊] Darren J. Creek,[‡] Kristian Kempe,[†] Pu Chun Ke[†*] and Thomas P. Davis[†,§*]*





Polymers	Mn_{NMR} (g mol^{-1})	Mn_{SEC} (g mol^{-1})	Đ
PEG	1980	4100	1.06
PEtOx	1970	3800	1.06

Figure S1. ^{1}H NMR spectra, SEC chromatograms, molecular weights and dispersities (Đ) for amine-functional PEG and PEtOx.

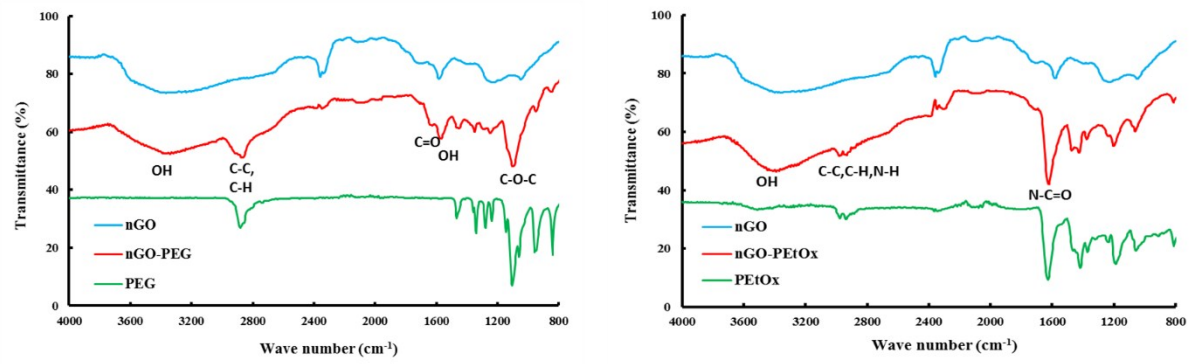
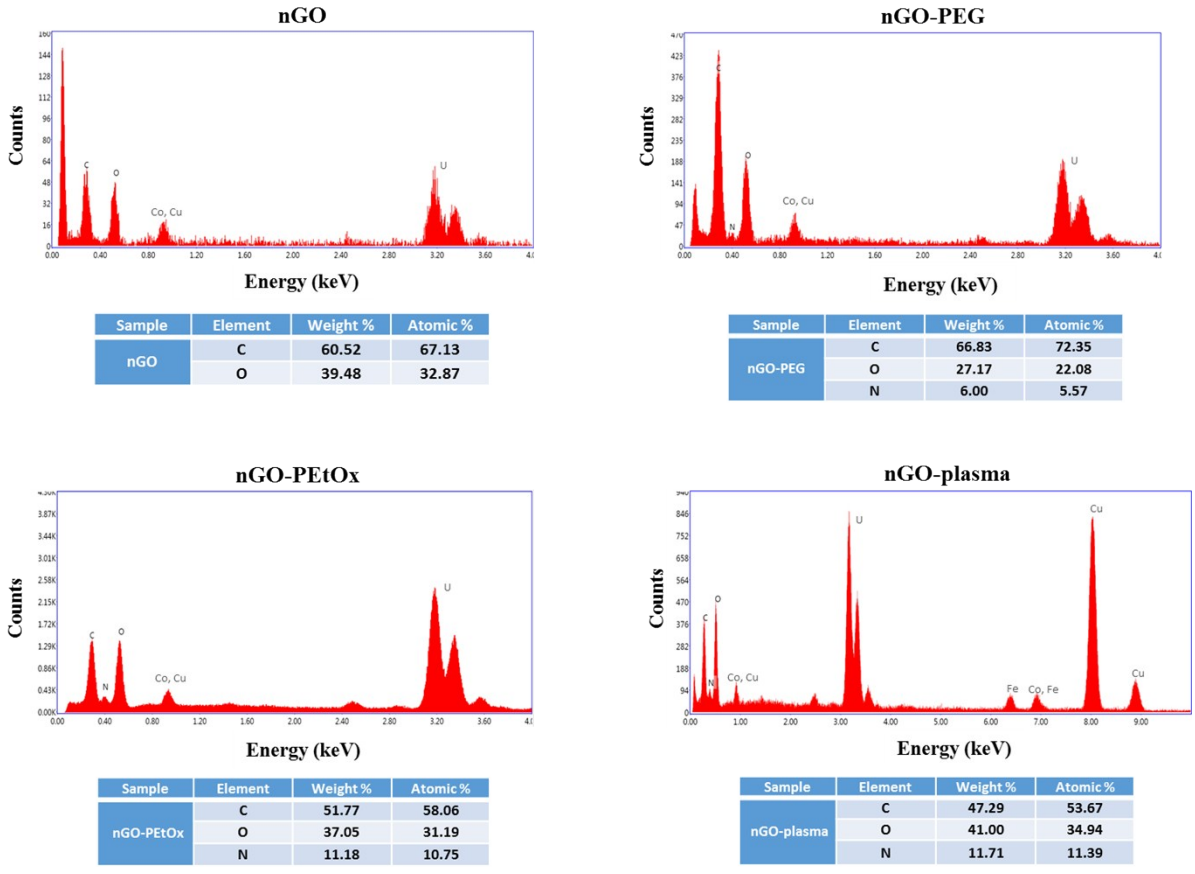


Figure S2. ATR-FTIR spectra of nGO, PEG and nGO-PEG (left); nGO, PEtOx and nGO-PEtOx (right).



Sample	Element	Weight %	Atomic %
nGO	C	60.52	67.13
	O	39.48	32.87

Sample	Element	Weight %	Atomic %
nGO-PEG	C	66.83	72.35
	O	27.17	22.08
	N	6.00	5.57

Sample	Element	Weight %	Atomic %
nGO-PEtOx	C	51.77	58.06
	O	37.05	31.19
	N	11.18	10.75

Sample	Element	Weight %	Atomic %
nGO-plasma	C	47.29	53.67
	O	41.00	34.94
	N	11.71	11.39

Figure S3. Energy-dispersive X-ray (EDX) spectra of nGO, nGO-PEG, nGO-PEtOx and nGO-plasma.

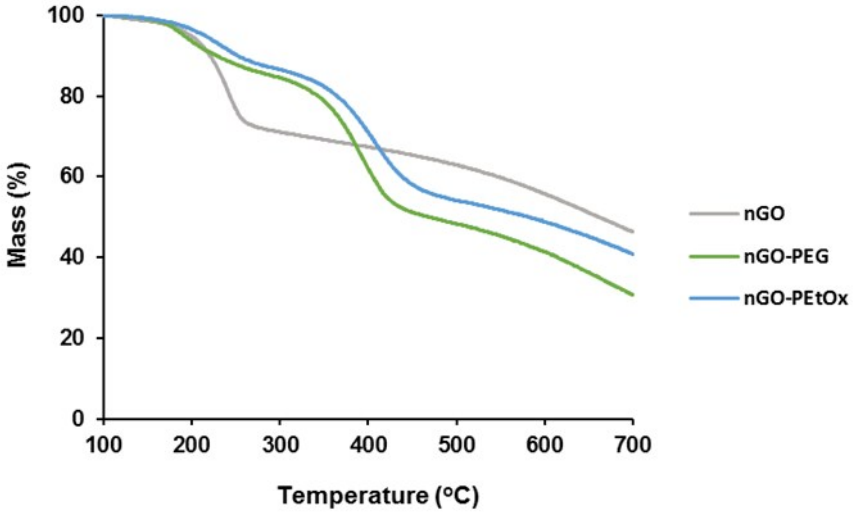


Figure S4. Thermogravimetric analysis (TGA) spectra of nGO, nGO-PEG and nGO-PEtOx.

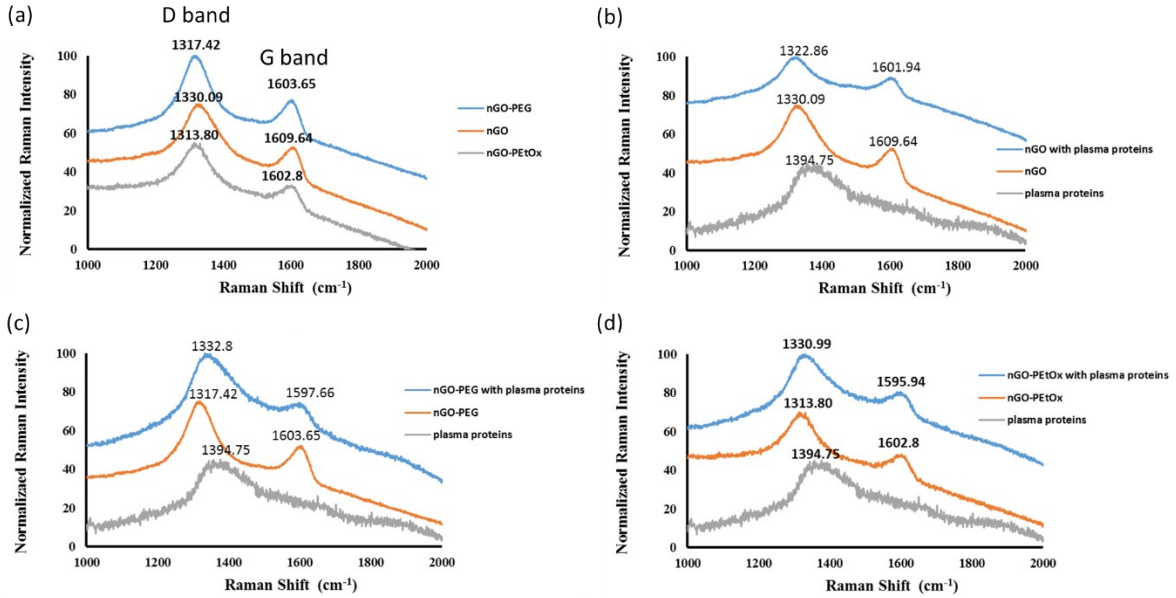


Figure S5. Raman spectra of (a) nGO, nGO-PEG and nGO-PEtOx (b) nGO, nGO with plasma proteins, and plasma proteins (c) nGO-PEG, nGO-PEG with plasma proteins, and plasma proteins (d) nGO-PEtOx, nGO-PEtOx with plasma proteins, and plasma proteins.

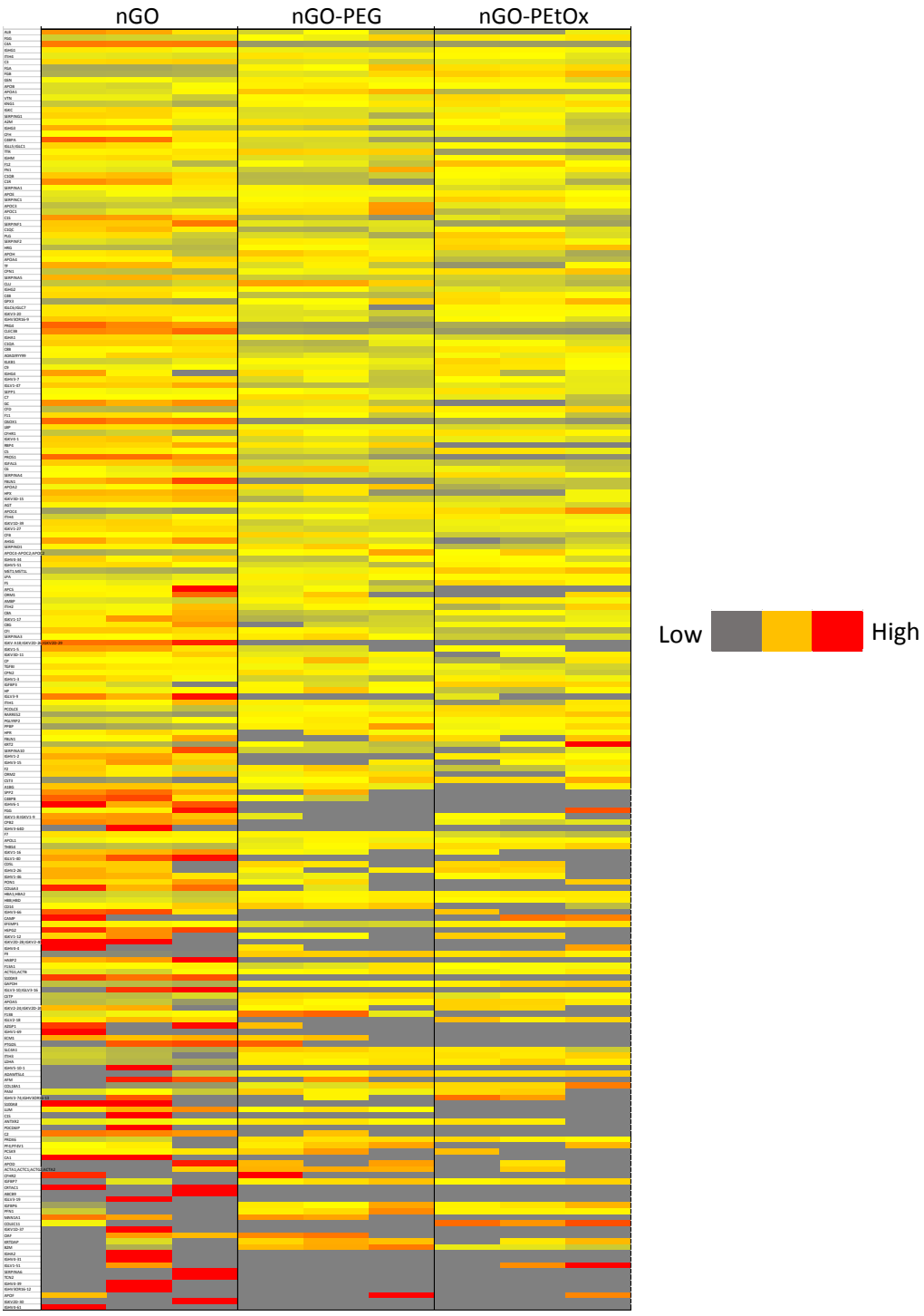


Figure S6. Heatmap of nGO, nGO-PEG and nGO-PEtOx coronal protein abundance levels plotted against normalized LFQ intensity.

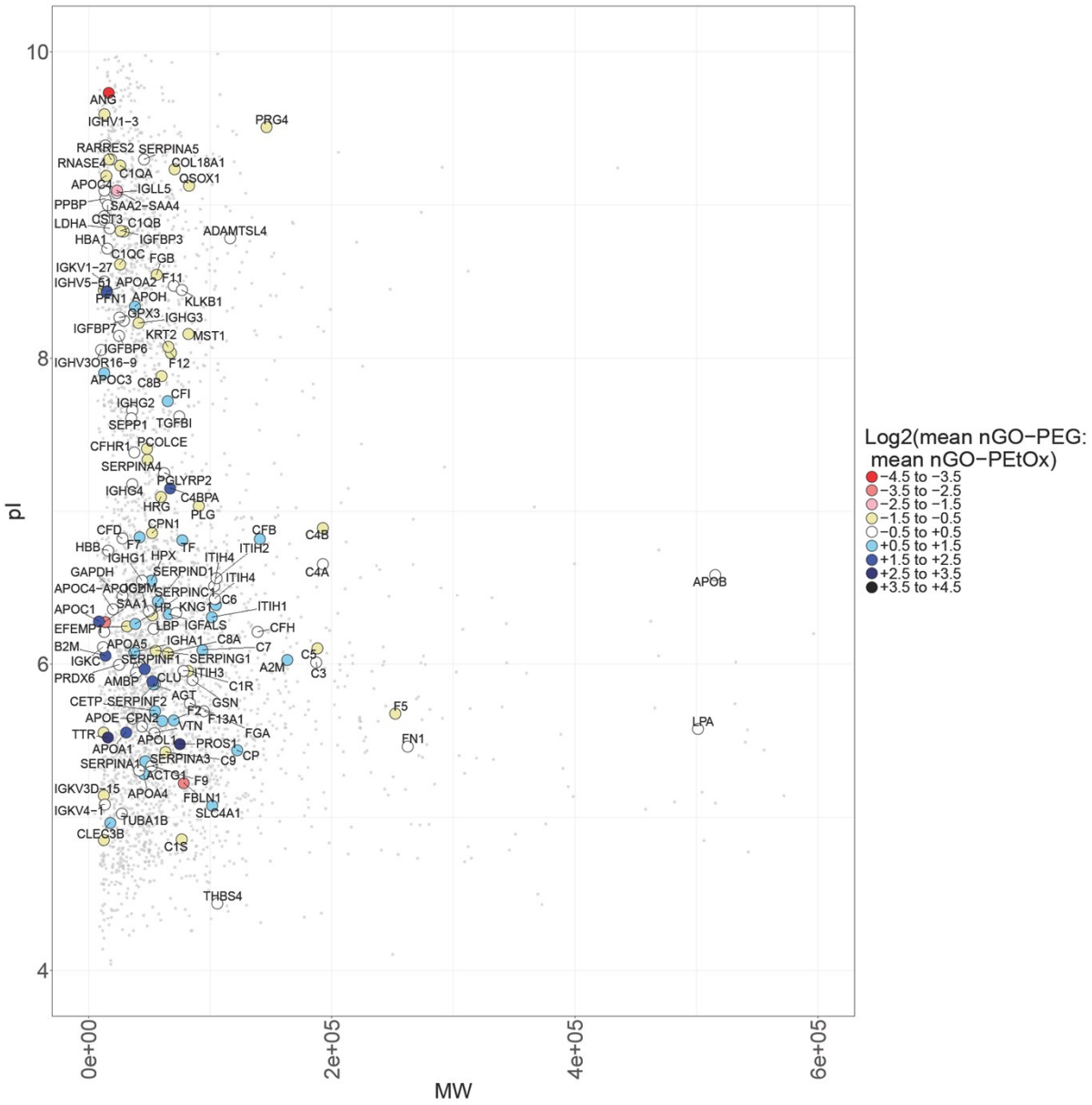


Figure S7. MW vs. pI distributions for corona proteins on nGO-PEG and -PEtOx are presented. Spot color indicates the ratio bin to which each protein belongs – based on $\log_2$ normalized ratio of mean LFQ (nGO-PEG):mean LFQ (nGO-PEtOx). The grey dots represent a non-glycosylated plasma background from Peptide Atlas (*20170801-063918*) without abundance information. Proteins are annotated with their respective gene names.

R script

1) https://creativecommons.org/licenses/by-sa/2.5/au/
2) Link to source for base R - https://www.r-project.org/
3) CRAN (Comprehensive R Archive Network) - https://cran.r-project.org/
4) Link to package "seqinr" - https://cran.r-project.org/web/packages/seqinr/index.html
5) Link to package "ggplot2" - https://cran.r-project.org/web/packages/ggplot2/index.html
6) Link to package "alakazam" - https://cran.r-project.org/web/packages/alakazam/index.html
7) Link to package "ggrepel" - https://cran.r-project.org/web/packages/ggrepel/index.html
8) Link to package "sqldf" - https://cran.r-project.org/web/packages/sqldf/index.html

Requirements:

1) Needs to be compiled as an R studio project.
2) Requires install of multiple packages (see libraries required and session information).
3) Requires both an "input" folder and "output" folder in the project directory.
4) The input folder should contain the following:
    a. proteinGroups text file
    b. query_guest_20170801-063917.csv (available from peptide atlas)
    c. uniprot-proteome_UP000005640.fasta.gz (available from uniprot.org)

NOTE: *.rmd file content begins on the next page.

NOTE: Session info is provided here:

```
## R version 3.3.3 (2017-03-06)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 7 x64 (build 7601) Service Pack 1
##
## locale:
## [1] LC_COLLATE=English_Australia.1252  LC_CTYPE=English_Australia.1252
## [3] LC_MONETARY=English_Australia.1252 LC_NUMERIC=C
## [5] LC_TIME=English_Australia.1252
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] sqldf_0.4-11   RSQLite_2.0    gsubfn_0.6-6   proto_1.0.0
## [5] seqinr_3.4-5   ggrepel_0.7.0  alakazam_0.2.8 ggplot2_2.2.1
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.12.13      plyr_1.8.4        bindr_0.1
##  [4] prettyunits_1.0.2 tools_3.3.3       progress_1.1.2
##  [7] bit_1.1-12        digest_0.6.12     memoise_1.1.0
## [10] evaluate_0.10.1   tibble_1.3.4      gtable_0.2.0
## [13] pkgconfig_2.0.1   rlang_0.1.4       igraph_1.1.2
## [16] DBI_0.7           yaml_2.1.14       bindrcpp_0.2
## [19] dplyr_0.7.4       stringr_1.2.0     knitr_1.17
## [22] hms_0.3           bit64_0.9-7       rprojroot_1.2
## [25] ade4_1.7-8        grid_3.3.3        glue_1.2.0
```

```
## [28] R6_2.2.2          tcltk_3.3.3       rmarkdown_1.7
## [31] blob_1.1.0        readr_1.1.1       magrittr_1.5
## [34] backports_1.1.1   scales_0.5.0      htmltools_0.3.6
## [37] assertthat_0.2.0  colorspace_1.3-2  labeling_0.3
## [40] stringi_1.1.5     lazyeval_0.2.1    munsell_0.4.3
## [43] chron_2.3-51
```

****

```
---
title: |
   | Electronic Supplementary Information
   | Human plasma proteome association and cytotoxicity of nano-graphene oxide grafted with stealth polyethylene glycol and poly-(2-ethyl-2-oxazoline)
output:
 html_document
---

***
# Document description
This *.html document contains the R script required to reproduce the figure elements in the main manuscript text - using the provided supplementary tables/files.

# R setup
```{r setup, include=TRUE}
knitr::opts_chunk$set(echo = TRUE, autodep = TRUE)
```
Data import, filtering and preliminary calculations
```{r import and processing}
###libraries required###
library(alakazam)
library(ggplot2)
library(ggrepel)
library(seqinr)
library(sqldf)

#####################################################################
####################Processing steps#################################
#####################################################################

#1. Data imported
data <- read.csv("./input/proteinGroups_re-run_151117.csv", sep=",", header=TRUE)

#2. Columns of interest extracted from dataframe
D <- data[,c("Gene.names", "Protein.names", "Protein.IDs", "Number.of.proteins", "Peptide.counts..unique.",
          "Unique.peptides", "Mol..weight..kDa.",
          "LFQ.intensity.PEG_exp2", "LFQ.intensity.PEG_exp3", "LFQ.intensity.PEG_exp4",
          "LFQ.intensity.POX_exp2", "LFQ.intensity.POX_exp3", "LFQ.intensity.POX_exp4")]

#3. Filters

#3.1 Proteins with >= 1 unique peptide retained
D <- subset(D, D$Unique.peptides >= 1)

#3.2 Rows with any zero values removed
#see https://stackoverflow.com/questions/11036989/replace-all-0-values-to-na
#see https://stackoverflow.com/questions/4862178/remove-rows-with-nas-missing-values-in-data-frame
```
```

```
D[D==0] <- NA
D <- na.omit(D)

#4. Multiple entries for gene names, protein IDs and protein names removed (first entry retained)
#see https://stackoverflow.com/questions/3703803/apply-strsplit-rowwise
D$Genes <- sapply(strsplit(as.character(D$Gene.names),";"), "[", 1)
D$Prot_ID <- sapply(strsplit(as.character(D$Protein.IDs),";"), "[", 1)
D$Prot_name <- sapply(strsplit(as.character(D$Protein.names),";"), "[", 1)

#5. LFQ intensities for each replicate (N=3) log transformed
D$LogPEG2 <- log2(D$LFQ.intensity.PEG_exp2)
D$LogPEG3 <- log2(D$LFQ.intensity.PEG_exp3)
D$LogPEG4 <- log2(D$LFQ.intensity.PEG_exp4)
D$LogPOX2 <- log2(D$LFQ.intensity.POX_exp2)
D$LogPOX3 <- log2(D$LFQ.intensity.POX_exp3)
D$LogPOX4 <- log2(D$LFQ.intensity.POX_exp4)

#6. Means and standard deviations for the log transformed replicates calculated
#see https://stackoverflow.com/questions/10945703/calculate-row-means-on-subset-of-columns
D$mean_logPEG <- rowMeans(D[,c("LogPEG2", "LogPEG3", "LogPEG4")])
D$mean_logPOX <- rowMeans(D[,c("LogPOX2", "LogPOX3", "LogPOX4")])
D$sd_logPEG <- sqrt(((D$LogPEG2-D$mean_logPEG)^2+(D$LogPEG3-D$mean_logPEG)^2+(D$LogPEG4-
D$mean_logPEG)^2)/2)
D$sd_logPOX <- sqrt(((D$LogPOX2-D$mean_logPOX)^2+(D$LogPOX3-D$mean_logPOX)^2+(D$LogPOX4-
D$mean_logPOX)^2)/2)

#7. PEG:PEtOx ratio calculated for mean LFQ values
D$meanPEG <- rowMeans(D[,c("LFQ.intensity.PEG_exp2","LFQ.intensity.PEG_exp3","LFQ.intensity.PEG_exp4")])
D$meanPOX <- rowMeans(D[,c("LFQ.intensity.POX_exp2","LFQ.intensity.POX_exp3","LFQ.intensity.POX_exp4")])
D$meanPEGPOX <- D$meanPEG/D$meanPOX
D$logmeanPEGPOX <- log2(D$meanPEGPOX)

#8. T-test for LFQ intensities (samples are not paired, equal variances assumed)
#see https://stackoverflow.com/questions/31596666/trying-to-perform-a-paired-t-test-for-each-row-and-count-all-rows-
where-p-value
pvalue <- rep(0,nrow(D))
for (i in 1:nrow(D)){pvalue[i] <-
t.test(D[i,c("LFQ.intensity.PEG_exp2","LFQ.intensity.PEG_exp3","LFQ.intensity.PEG_exp4")],
D[i,c("LFQ.intensity.POX_exp2","LFQ.intensity.POX_exp3","LFQ.intensity.POX_exp4")],
paired=FALSE,var.equal=TRUE)$p.value}
D <- cbind(D,pvalue)

#9. Subset and check
nrow(D)
Dp <- subset(D, D$pvalue < 0.05)
nrow(Dp)

#10. p-value corrections
#see https://stat.ethz.ch/R-manual/R-devel/library/stats/html/p.adjust.html
BF <- p.adjust(pvalue, n = length(pvalue), method = "bonferroni")
FDR <- p.adjust(pvalue, n = length(pvalue), method = "fdr")

#11. Create combined dataframe
D2 <- cbind(D[,c("Genes", "Prot_ID", "Prot_name", "meanPEGPOX")], pvalue, BF, FDR)
```

Protein sequence-based calculations and plasma background

```{r protein calculations and plasma background}
```

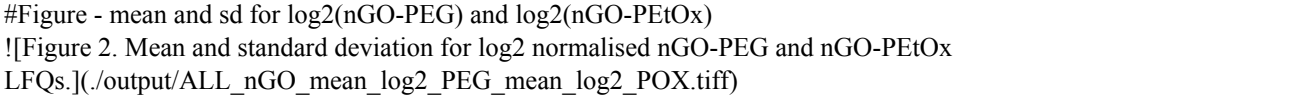
```
#################################################################
###################GetAnnotSeqFASTA#############################
#################################################################
###function to combine protein name, accession, gene name#############
#############and sequence from a *.fasta file#####################
#################################################################

GetAnnotSeqFASTA <- function(x){
  UP_sequences <- unlist(getSequence(x, as.string=TRUE))
  Annotations <- unlist(getAnnot(x, as.string=TRUE))
  ProtName_1 <- strsplit(Annotations, "AN ", fixed=TRUE)
  ProtName_2 <- rapply(ProtName_1, function(a) a[2])
  ProtName_3 <- strsplit(ProtName_2, " OS", fixed=TRUE)
  UP_prot_names <- rapply(ProtName_3, function(a) a[1])
  Genes_1 <- strsplit(Annotations, "GN=", fixed=TRUE)
  Genes_2 <- rapply(Genes_1, function(a) a[2])
  Genes_3 <- strsplit(Genes_2, " ", fixed=TRUE)
  UP_gene_names <- rapply(Genes_3, function(a) a[1])
  Prots_1 <- strsplit(Annotations, "|", fixed=TRUE)
  UP_prot_ID <- rapply(Prots_1, function(a) a[2])
  data <- data.frame(UP_prot_ID, UP_prot_names, UP_gene_names, UP_sequences)
  return(data)
}

###load and process uniprot *.fasta file for Homo sapiens###
UP <- read.fasta(file="./input/uniprot-proteome_UP000005640.fasta.gz", seqtype="AA", as.string=TRUE)
dataUP <- GetAnnotSeqFASTA(UP)

###################################
###Matching to obtain sequences#######
###################################

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

###string split extracted sequences and calculated GRAVY, pI and MW###
split_SEQ <- strsplit(as.character(extract$UP_sequences), "")
pI <- unlist(lapply(split_SEQ, computePI))
MW <- unlist(lapply(split_SEQ, pmw))
extract$GRAVY <- gravy(extract$UP_sequences)
extract <- cbind(extract, pI, MW)

###Note, entries that contain selenocysteine (U)###
######are not included in the final figures########

nrow(D)
nrow(extract)

#################################################################################################
###############data obtained from peptideATLAS using the NextProtChromMapping##################
########https://db.systemsbiology.net/sbeams/cgi/PeptideAtlas/GetNextProtChromMapping###########
######################[Annotate this Resultset] Name: " (20170801-063918)#######################
###URL to recall this result set: https://db.systemsbiology.net/sbeams/cgi/shortURL?key=cga2kglb#########
###URL to re-execute this query: https://db.systemsbiology.net/sbeams/cgi/shortURL?key=5ag17srw########
########################################Query Time: 0.13 s ######################################
#################################################################################################
```

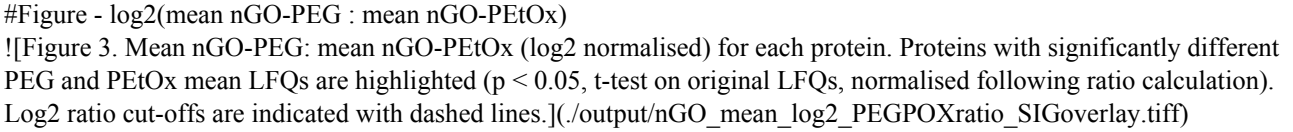
```
###load peptideATLAS data -- match accessions across plasma and uniprot data###
PLASMA <- read.csv(file="./input/query_guest_20170801-063917.csv", sep=",", header=TRUE)
plasma <- sqldf("select * from dataUP inner join PLASMA on PLASMA.nextprot_accession = dataUP.UP_prot_ID")

###string split plasma sequences and calculate GRAVY, pI and MW###
split_SEQ_plasma <- strsplit(as.character(plasma$UP_sequences), "")
pI <- unlist(lapply(split_SEQ_plasma, computePI))
MW <- unlist(lapply(split_SEQ_plasma, pmw))
plasma$GRAVY <- gravy(plasma$UP_sequences)
plasma_FINAL <- cbind(plasma, pI, MW)

###export protein lists###
write.table(extract, "./output/processed_data.txt", sep="\t")
write.table(plasma_FINAL, "./output/plasma.txt", sep="\t")
```

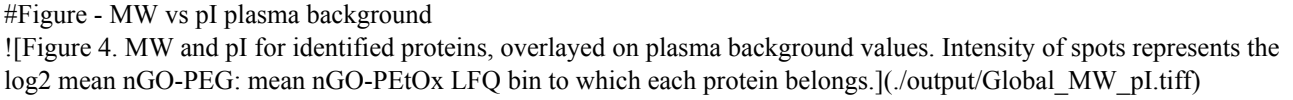
Code for plots
```{r lfq plots}

###theme settings###
theme <- theme(legend.text = element_text(size = 15), legend.title=element_text(size=20),
axis.title=element_text(size=20), legend.key.size = unit(0.8, "cm"), axis.text.x = element_text(angle=90, hjust=1,
vjust=0.5, size=15), axis.text.y = element_text(size=10))

theme2 <- theme(legend.text = element_text(size = 15), legend.title=element_text(size=20),
axis.title=element_text(size=15), legend.key.size = unit(0.8, "cm"), axis.text.x = element_text(vjust=0.5, size=15),
axis.text.y = element_text(size=10))

theme3 <- theme(legend.text = element_text(size = 25), legend.title=element_text(size=35),
axis.title=element_text(size=35), legend.key.size = unit(0.8, "cm"), axis.text.x = element_text(angle=90, hjust=1,
vjust=0.5, size=35), axis.text.y = element_text(size=35))

a <- ggplot(D2, aes(reorder(Prot_ID, pvalue), pvalue)) + geom_point(shape=21, fill="blue", size=2, alpha=0.5)
a <- a + coord_flip()
a <- a + geom_point(aes(Prot_ID, BF),shape=21,  size=2, fill="red",alpha=0.5)
a <- a + geom_point(aes(Prot_ID, FDR),shape=21,  size=2, fill="green",alpha=0.5)
a <- a + xlab("Protein ID") + ylab("p-values") + theme_bw() + theme
tiff("./output/pvalue_corrections.tiff", width=800, height=1200)
plot(a)
dev.off()
pdf("./output/pvalue_corrections.pdf", width=10, height=18)
plot(a)
dev.off()
b <- ggplot(D, aes(reorder(Prot_ID, mean_logPEG), mean_logPEG)) + geom_point(col="blue", alpha=0.7)
b <- b + geom_point(aes(Prot_ID, mean_logPOX), col="red")
b <- b + coord_flip()
b <- b + geom_errorbar(aes(ymin=mean_logPEG-sd_logPEG, ymax=mean_logPEG+sd_logPEG), col="blue",
alpha=0.7, width=.1)
b <- b + geom_errorbar(aes(ymin=mean_logPOX-sd_logPOX, ymax=mean_logPOX+sd_logPOX), col="red",
alpha=0.7, width=.1)
b <- b + xlab("Protein ID") + ylab("Mean log2(LFQs)") + theme_bw() + theme2
tiff("./output/ALL_nGO_mean_log2_PEG_mean_log2_POX.tiff", width=800, height=1200)
plot(b)
dev.off()
pdf("./output/ALL_nGO_mean_log2_PEG_mean_log2_POX.pdf", width=10, height=18)
plot(b)
dev.off()
```

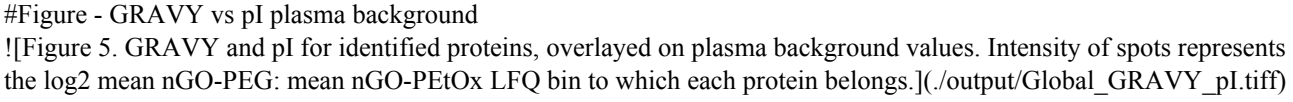
```
c <- ggplot(D, aes(reorder(Prot_ID, logmeanPEGPOX), logmeanPEGPOX)) + geom_point(size=3, col="gray")
c <- c + geom_point(data=Dp, aes(Prot_ID, logmeanPEGPOX), size=3, fill="red", shape=21)
c <- c + coord_flip()
c <- c + geom_hline(aes(yintercept = log2(0.5)), col="black", linetype="dashed", size=0.4)
c <- c + geom_hline(aes(yintercept = log2(2.0)), col="black", linetype="dashed", size=0.4)
c <- c + xlab("Protein ID") + ylab("Log2(mean nGO-PEG: mean nGO-PEtOx)") + theme_bw() + theme2
tiff("./output/nGO_mean_log2_PEGPOXratio_SIGoverlay.tiff", width=600, height=1600)
plot(c)
dev.off()
pdf("./output/nGO_mean_log2_PEGPOXratio_SIGoverlay.pdf", width=6, height=18)
plot(c)
dev.off()

###temp dataframe to bin logMEAN values###
#see https://gis.stackexchange.com/questions/178597/how-to-create-a-continuous-scale-with-distinct-custom-color-and-value-breaks-wit
extract$ID <- cut(extract$logmeanPEGPOX, breaks=c(-4.5,-3.5,-2.5,-1.5,-0.5,0.5,1.5,2.5,3.5,4.5), labels=c("-4.5 to -3.5","-3.5 to -2.5","-2.5 to -1.5","-1.5 to -0.5","-0.5 to +0.5","+0.5 to +1.5", "+1.5 to +2.5", "+2.5 to +3.5", "+3.5 to +4.5"))

###set colour scale for plotting of bins###
#see http://sape.inf.usi.ch/quick-reference/ggplot2/colour - used as colour reference
COL <- c("red", "lightcoral", "pink1", "palegoldenrod", "white", "skyblue", "blue", "navyblue", "black")

###Global comparison of MW and pI###
d <- ggplot(plasma_FINAL, aes(MW, pI)) + geom_point(col="gray", alpha=0.5)
d <- d + geom_point(data=extract, aes(MW, pI, fill=ID), pch=21, size=8,col="black")
d <- d + scale_fill_manual(values = COL, drop=FALSE)
d <- d + geom_text_repel(data=extract, aes(MW, pI, label=Genes), size=7)
d <- d + guides(fill=guide_legend(title="Log2(mean nGO-PEG: \n mean nGO-PEtOx)"))
d <- d + scale_y_continuous(limits=c(4,10)) + scale_x_continuous(limits=c(0,600000))
d <- d + xlab("MW") + ylab("pI") + theme_bw() + theme3
tiff("./output/Global_MW_pI.tiff", width=1600, height=1600)
plot(d)
dev.off()
pdf("./output/Global_MW_pI.pdf", width=25, height=25)
plot(d)
dev.off()

###Global comparison of GRAVY and pI###
e <- ggplot(plasma_FINAL, aes(GRAVY, pI)) + geom_point(col="gray", alpha=0.5)
e <- e + geom_point(data=extract, aes(GRAVY, pI, fill=ID), pch=21, size=8,col="black")
e <- e + scale_fill_manual(values = COL, drop=FALSE)
e <- e + geom_text_repel(data=extract, aes(GRAVY, pI, label=Genes), size=7)
e <- e + guides(fill=guide_legend(title="Log2(mean nGO-PEG: \n mean nGO-PEtOx)"))
e <- e + scale_y_continuous(limits=c(4,10)) + scale_x_continuous(limits=c(-1.25,0.5))
e <- e + xlab("GRAVY") + ylab("pI") + theme_bw() + theme3
tiff("./output/Global_GRAVY_pI.tiff", width=1600, height=1600)
plot(e)
dev.off()
pdf("./output/Global_GRAVY_pI.pdf", width=25, height=25)
plot(e)
dev.off()

```

#Figure - p-values and corrections
![Figure 1. p-value corrections for t-tests comparing mean of LFQs for proteins detected in both nGO-PEG and nGO-PEtOx samples.](./output/pvalue_corrections.tiff)

#Figure - mean and sd for log2(nGO-PEG) and log2(nGO-PEtOx)


#Figure - log2(mean nGO-PEG : mean nGO-PEtOx)


#Figure - MW vs pI plasma background


#Figure - GRAVY vs pI plasma background


Session information
```{r sessionInfo}
sessionInfo()
```

Package citations (with links):

Wickham H (2009). ggplot2: Elegant Graphics for Data Analysis. Springer-Verlag New York. http://www.springer.com/us/book/9780387981413

Gupta N, Vander Heiden J, Uduman M, Gadala-Maria D, Yaari G and Kleinstein S (2015). “Change-O: a toolkit for analyzing large-scale B cell immunoglobulin repertoire sequencing data.” _Bioinformatics_, pp. 1-3. doi: 10.1093/bioinformatics/btv359 (URL: http://doi.org/10.1093/bioinformatics/btv359).

Slowikowski K (2017). ggrepel: Repulsive Text and Label Geoms for 'ggplot2'. R package version 0.7.0. https://CRAN.R-project.org/package=ggrepel

Charif D, Lobry J R (2007). SeqinR 1.0-2: A Contributed Package to the R Project for Statistical Computing Devoted to Biological Sequences Retrieval and Analysis. Structural Approaches to Sequence Evolution, ppp. 207-232. https://link.springer.com/chapter/10.1007/978-3-540-35306-5_10

Grothendieck G (2017). sqldf: Manipulate R Data Frames Using SQL. R package version 0.4-11. http://CRAN.R-project.org/package=sqldf