

# HTG-validation\_RS-GGI-SET-Pen.R

t

2024-08-07

```
# HEADER #####
#
# Version: 2024-07-19
#
# Comparison of signature scores in TCGA-BRCA-RNA-Seq based on
# complete gene list and the subset available in the HTG-Panel
# for the following signature scores:
#
#
# - oncototypeDX/Recurrence Score surrogate: adapted from genefu (PMID 26607490)
#       based on Paik 2004 (PMID 15591335)
# - GGI (Genomic Grade Index): genefu version with 112 genes (PMID 26607490)
#       based on Sotiriou 2006 (PMID 16478745)
# - SET-Index: Robust 18-gene predictor from Sinn 2019 (PMID 31231679 )
#       based on Symmans 2010 (PMID 20697068)
# - Pen355: n355 DEG from paired analysis of 1080 paired pre/post Penelope samples
#       (Different scores for subclusters of the 355 genes)
#
#
#
#
# SETUP #####
```

```
Sys.setenv(lang = "en_US")
```

Install required packages if missing -----

```
# Package names from CRAN
packs <- c("dplyr")

# Install packages not yet installed
installed_packages <- packs %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages])
}

# Package names from Bioconductor
bcpacks <- c("genefu", "cBioPortalData")

# Install bc-packages if not yet installed from Bioconductor
installed_packages <- bcpacks %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
  BiocManager::install(packages[!installed_packages])
}
```

*Load required packages* —————

```
invisible(lapply(packs, library, character.only = TRUE))

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##   filter, lag

## The following objects are masked from 'package:base':
##
##   intersect, setdiff, setequal, union

invisible(lapply(bcpacks, library, character.only = TRUE))

## Loading required package: survcomp
## Loading required package: survival
## Loading required package: prodlim
## Warning: package 'prodlim' was built under R version 4.3.1
## Loading required package: biomaRt
## Loading required package: iC10
## Warning: package 'iC10' was built under R version 4.3.1
## Loading required package: pamr
## Warning: package 'pamr' was built under R version 4.3.1
## Loading required package: cluster
## Loading required package: impute
## Loading required package: iC10TrainingData
## Loading required package: AIMS
## Loading required package: e1071
## Loading required package: Biobase
## Loading required package: BiocGenerics
##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:dplyr':
##
##   combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##   IQR, mad, sd, var, xtabs
```

```

## The following objects are masked from 'package:base':
##
##     anyDuplicated, aperm, append, as.data.frame, basename, cbind,
##     colnames, dirname, do.call, duplicated, eval, evalq, Filter, Find,
##     get, grep, grepl, intersect, is.unsorted, lapply, Map, mapply,
##     match, mget, order, paste, pmax, pmax.int, pmin, pmin.int,
##     Position, rank, rbind, Reduce, rownames, sapply, setdiff, sort,
##     table, tapply, union, unique, unsplit, which.max, which.min

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Loading required package: AnVIL

## Loading required package: MultiAssayExperiment

## Loading required package: SummarizedExperiment

## Loading required package: MatrixGenerics

## Warning: package 'MatrixGenerics' was built under R version 4.3.1

## Loading required package: matrixStats

## Warning: package 'matrixStats' was built under R version 4.3.1

##
## Attaching package: 'matrixStats'

## The following objects are masked from 'package:Biobase':
##
##     anyMissing, rowMedians

## The following object is masked from 'package:dplyr':
##
##     count

##
## Attaching package: 'MatrixGenerics'

## The following objects are masked from 'package:matrixStats':
##
##     colAlls, colAnyNAs, colAnys, colAvgPerRowSet, colCollapse,
##     colCounts, colCummaxs, colCummins, colCumprods, colCumsums,
##     colDiffs, colIQRDiffs, colIQRs, colLogSumExps, colMadDiffs,
##     colMads, colMaxs, colMeans2, colMedians, colMins, colOrderStats,
##     colProds, colQuantiles, colRanges, colRanks, colSdDiffs, colSds,
##     colSums2, colTabulates, colVarDiffs, colVars, colWeightedMads,
##     colWeightedMeans, colWeightedMedians, colWeightedSds,
##     colWeightedVars, rowAlls, rowAnyNAs, rowAnys, rowAvgPerColSet,
##     rowCollapse, rowCounts, rowCummaxs, rowCummins, rowCumprods,
##     rowCumsums, rowDiffs, rowIQRDiffs, rowIQRs, rowLogSumExps,
##     rowMadDiffs, rowMads, rowMaxs, rowMeans2, rowMedians, rowMins,
##     rowOrderStats, rowProds, rowQuantiles, rowRanges, rowRanks,
##     rowSdDiffs, rowSds, rowSums2, rowTabulates, rowVarDiffs, rowVars,

```

```
##      rowWeightedMads, rowWeightedMeans, rowWeightedMedians,
##      rowWeightedSds, rowWeightedVars

## The following object is masked from 'package:Biobase':
##
##      rowMedians

## Loading required package: GenomicRanges

## Loading required package: stats4

## Loading required package: S4Vectors

##
## Attaching package: 'S4Vectors'

## The following objects are masked from 'package:dplyr':
##
##      first, rename

## The following object is masked from 'package:utils':
##
##      findMatches

## The following objects are masked from 'package:base':
##
##      expand.grid, I, unname

## Loading required package: IRanges

## Warning: package 'IRanges' was built under R version 4.3.1

##
## Attaching package: 'IRanges'

## The following objects are masked from 'package:dplyr':
##
##      collapse, desc, slice

## The following object is masked from 'package:grDevices':
##
##      windows

## Loading required package: GenomeInfoDb

## Warning: package 'GenomeInfoDb' was built under R version 4.3.1
```

```

# FUNCTION Definitions #####
# *****
#
# Function oncotypedxHTG #####
#   with "CTSL2" updated to "CTSV"
#   excluding "GRB7"
#
#
# The function oncotypedxHTG is a modified version
# of the oncotypedx function from library genefu
#
# In function oncotypedxHTG the gene symbol "CTSL2" is replaced by the
# current HUGO symbol "CTSV".
#
# In addition the gene "GRB7" (which is not available in HTG-Panel) is
# left out of the algorithm in oncotypedxHTG.
#
# Moreover, the variable "sig.oncotypedx" is replaced by "sig.oncotypedx.htg"
# --> within this new table "sig.oncotypedx.htg" the gene symbol "CTSL2" is
# updated to the current HUGO symbol "CTSV" and the row with "GRB7" has
# been removed.
#
# The scaling and clipping of rsu values in the original function
# from genefu package has been modified in order to adapt the
# distribution of Recurrence Score values from RNA-Seq
# to those from clinical RS data (PMID 32565552)

oncotypedxHTG <- function (data, annot, do.mapping = FALSE, mapping, verbose = FALSE)
{
  sig2 <- sig.oncotypedx.htg[sig.oncotypedx.htg[, "group"] != "reference",
                             , drop = FALSE]
  dimnames(sig2)[[1]] <- sig2[, "symbol"]
  gt <- nrow(sig2)
  if (do.mapping) {
    gid1 <- as.numeric(as.character(sig2[, "EntrezGene.ID"]))
    names(gid1) <- dimnames(sig2)[[1]]
    gid2 <- as.numeric(as.character(annot[, "EntrezGene.ID"]))
    names(gid2) <- dimnames(annot)[[1]]
    rm.ix <- is.na(gid1) | duplicated(gid1)
    gid1 <- gid1[!rm.ix]
    rr <- geneid.map(geneid1 = gid2, data1 = data, geneid2 = gid1,
                    verbose = FALSE)
    gm <- length(rr$geneid2)
    mymapping <- c(mapped = gm, total = gt)
    if (length(rr$geneid1) != gt) {
      res <- rep(NA, nrow(data))
      names(res) <- dimnames(data)[[1]]
      warning(sprintf("Probe candidates: %i/%i", gm, gt),
              "\nIncomplete overlap between the gene signature EntrezGene.IDs",
              " and the EntrezGene.ID column of annot... Returning all NAs.")
      return(list(score = res, risk = res, mapping = mymapping,
                  probe = NA))
    }
  }
  gid1 <- rr$geneid2

```

```

gid2 <- rr$geneid1
data <- rr$data1
myprobe <- cbind(probe = names(gid1), EntrezGene.ID = gid1,
                  new.probe = names(gid2))
dimnames(data)[[2]] <- names(gid2) <- names(gid1)
}else {
  myprobe <- NA
  data <- data[, intersect(dimnames(sig2)[[1]], dimnames(data)[[2]])]
  #data <- data[, intersect(sig2$symbol, dimnames(data)[[2]])]
  gm <- ncol(data)
  mymapping <- c(mapped = gm, total = gt)
  if (nrow(sig2) != ncol(data)) {
    res <- rep(NA, nrow(data))
    names(res) <- dimnames(data)[[1]]
    warning(sprintf("Probe candidates: %i/%i", gm, gt),
              "\nIncomplete overlap between the gene signature EntrezGene.IDs",
              " and the colnames of data... Returning all NAs.")
    return(list(score = res, risk = res, mapping = mymapping,
                probe = myprobe))
  }
}
dimnames(data)[[2]] <- dimnames(sig2)[[1]] <- sig2[, "symbol"]
data <- apply(data, 2, function(x) {
  xx <- (x - min(x, na.rm = TRUE))/(max(x, na.rm = TRUE) -
                                     min(x, na.rm = TRUE))

  return(xx * 15)
})
cc.ix <- complete.cases(data)
rs <- rs.unscaled <- rs.risk <- NULL
for (i in 1:nrow(data)) {
  if (cc.ix[i]) {
    # grb7.gs <- 0.9 * data[i, "GRB7"] + 0.1 * data[i, "ERBB2"]
    grb7.gs <- 0.7 * data[i, "ERBB2"] # grb7.gs replaced by just ERBB2 value
    # 0.7 represents the median expression of GRB7 vs ERBB2 in TCGA-ERpos
    if (grb7.gs < 8) {
      grb7.gs <- 8
    }
    er.gs <- (0.8 * data[i, "ESR1"] + 1.2 * data[i, "PGR"] +
              data[i, "BCL2"] + data[i, "SCUBE2"])/4
    proliferation.gs <- (data[i, "BIRC5"] + data[i, "MKI67"] +
                          data[i, "MYBL2"] + data[i, "CCNB1"] +
                          data[i, "AURKA"])/5
    if (proliferation.gs < 6.5) {
      proliferation.gs <- 6.5
    }
    invasion.gs <- (data[i, "CTSV"] + data[i, "MMP11"])/2
    rsu <- 0.47 * (grb7.gs) - 0.34 * (er.gs) + 1.04 *
      (proliferation.gs) + 0.1 * (invasion.gs) + 0.05 *
      data[i, "CD68"] - 0.08 * data[i, "GSTM1"] - 0.07 *
      data[i, "BAG1"]
    rsu2 <- rsu
    rsu <- rsu * 4 - 18 # adapted from distribution from PMID 32565552
    if (rsu < 11) {
      rsr <- 0
    }
  }
}

```

```

    if (rsu >= 11 & rsu < 26) {
      rsr <- 0.5
    }
    if (rsu >= 26) {
      rsr <- 1
    }
  }
  else {
    rsu <- rsr <- rsu2 <- NA
  }
  rs.unscaled <- c(rs.unscaled, rsu2)
  rs <- c(rs, rsu)
  rsrisk <- c(rsrisk, rsr)
}
names(rs) <- names(rs.unscaled) <- names(rsrisk) <- dimnames(data)[[1]]
return(list(score = rs, risk = rsrisk, mapping = mymapping,
            probe = myprobe))
}

```

```

# Function oncotypedxCTSV with "CTSL2" updated to "CTSV" ####
#
#
# The function oncotypedxCTSV is a modified version
# of the oncotypedx function from library genefu
#
# In function oncotypedxCTSV the gene symbol "CTSL2" is replaced by the
# current HUGO symbol "CTSV". Moreover, the variable "sig.oncotypedx"
# is replaced by "sig.oncotypedx.new"
# --> within this new table "sig.oncotypedx.new" the gene symbol "CTSL2" is
# updated to the current HUGO symbol "CTSV".
#
# The scaling and clipping of rsu values in the original function
# from genefu package has been modified in order to adapt the
# distribution of Recurrence Score values from RNA-Seq
# to those from clinical RS data (PMID 32565552)

oncotypedxCTSV <- function (data, annot, do.mapping = FALSE, mapping, verbose = FALSE)
{
  sig2 <- sig.oncotypedx.new[sig.oncotypedx.new[, "group"] != "reference",
                             , drop = FALSE]
  dimnames(sig2)[[1]] <- sig2[, "symbol"]
  gt <- nrow(sig2)
  if (do.mapping) {
    gid1 <- as.numeric(as.character(sig2[, "EntrezGene.ID"]))
    names(gid1) <- dimnames(sig2)[[1]]
    gid2 <- as.numeric(as.character(annot[, "EntrezGene.ID"]))
    names(gid2) <- dimnames(annot)[[1]]
    rm.ix <- is.na(gid1) | duplicated(gid1)
    gid1 <- gid1[!rm.ix]
    rr <- geneid.map(geneid1 = gid2, data1 = data, geneid2 = gid1,
                    verbose = FALSE)
    gm <- length(rr$geneid2)
    mymapping <- c(mapped = gm, total = gt)
    if (length(rr$geneid1) != gt) {
      res <- rep(NA, nrow(data))
      names(res) <- dimnames(data)[[1]]
      warning(sprintf("Probe candidates: %i/%i", gm, gt),
              "\nIncomplete overlap between the gene signature EntrezGene.IDs",
              " and the EntrezGene.ID column of annot... Returning all NAs.")
      return(list(score = res, risk = res, mapping = mymapping,
                  probe = NA))
    }
    gid1 <- rr$geneid2
    gid2 <- rr$geneid1
    data <- rr$data1
    myprobe <- cbind(probe = names(gid1), EntrezGene.ID = gid1,
                    new.probe = names(gid2))
    dimnames(data)[[2]] <- names(gid2) <- names(gid1)
  } else {
    myprobe <- NA
    data <- data[, intersect(dimnames(sig2)[[1]], dimnames(data)[[2]])]
    #data <- data[, intersect(sig2$symbol, dimnames(data)[[2]])]
    gm <- ncol(data)
    mymapping <- c(mapped = gm, total = gt)
  }
}

```



```

if (nrow(sig2) != ncol(data)) {
  res <- rep(NA, nrow(data))
  names(res) <- dimnames(data)[[1]]
  warning(sprintf("Probe candidates: %i/%i", gm, gt),
    "\nIncomplete overlap between the gene signature EntrezGene.IDs",
    " and the colnames of data... Returning all NAs.")
  return(list(score = res, risk = res, mapping = mymapping,
    probe = myprobe))
}
}
dimnames(data)[[2]] <- dimnames(sig2)[[1]] <- sig2[, "symbol"]
data <- apply(data, 2, function(x) {
  xx <- (x - min(x, na.rm = TRUE)) / (max(x, na.rm = TRUE) -
    min(x, na.rm = TRUE))
  return(xx * 15)
})
cc.ix <- complete.cases(data)
rs <- rs.unscaled <- rsrisk <- NULL
for (i in 1:nrow(data)) {
  if (cc.ix[i]) {
    grb7.gs <- 0.9 * data[i, "GRB7"] + 0.1 * data[i, "ERBB2"]
    if (grb7.gs < 8) {
      grb7.gs <- 8
    }
    er.gs <- (0.8 * data[i, "ESR1"] + 1.2 * data[i, "PGR"] +
      data[i, "BCL2"] + data[i, "SCUBE2"]) / 4
    proliferation.gs <- (data[i, "BIRC5"] + data[i, "MKI67"] +
      data[i, "MYBL2"] + data[i, "CCNB1"] +
      data[i, "AURKA"]) / 5
    if (proliferation.gs < 6.5) {
      proliferation.gs <- 6.5
    }
    invasion.gs <- (data[i, "CTSV"] + data[i, "MMP11"]) / 2
    rsu <- 0.47 * (grb7.gs) - 0.34 * (er.gs) + 1.04 *
      (proliferation.gs) + 0.1 * (invasion.gs) + 0.05 *
      data[i, "CD68"] - 0.08 * data[i, "GSTM1"] - 0.07 *
      data[i, "BAG1"]
    rsu2 <- rsu
    rsu <- rsu * 4 - 18 # adapted from distribution from PMID 32565552
    if (rsu < 11) {
      rsr <- 0
    }
    if (rsu >= 11 & rsu < 26) {
      rsr <- 0.5
    }
    if (rsu >= 26) {
      rsr <- 1
    }
  }
  else {
    rsu <- rsr <- rsu2 <- NA
  }
  rs.unscaled <- c(rs.unscaled, rsu2)
  rs <- c(rs, rsu)
  rsrisk <- c(rsrisk, rsr)
}

```

```
}  
names(rs) <- names(rs.unscaled) <- names(rsrisk) <- dimnames(data)[[1]]  
return(list(score = rs, risk = rsrisk, mapping = mymapping,  
           probe = myprobe))  
}  
  
# *****
```

```

# Function tcgaRseqGenelist #####
#
#
# The function tcgaRseqGenelist obtains RNA-Seq data of a provided genelist
# from TCGA using the cBioPortal access tools.
#
# We apply the cBioPortalData package to access data from the cBIO Portal
# at www.cbioportal.org
# This will allow to download RNA-Seq data from the TCGA-BRCA cohort.

library(cBioPortalData)
# First we setup some parameters for the cBioportal-access
# Define api
cbio <- cBioPortal()

## Warning in .service_validate_md5sum(api_reference_url, api_reference_md5sum, : service
version differs from validated version
##      service url: https://www.cbioportal.org/api/v2/api-docs
##      observed md5sum: 7314de5c5e8056e4e07b411b3e5a0cb9
##      expected md5sum: 07ceb76cc5afcf54a9cf2e1a689b18f7

# Function definition:
# (genelist is a vector of gene symbols)

tcgaRseqGenelist <- function (genelist) {
  # Download BRCA RNA-Seq data for this genelist from cBioPortal
  # as a "MultiAssayExperiment" brca_rnaseq
  brca_rnaseq <- cBioPortalData(
    api = cbio,
    studyId = "brca_tcga",
    genes = genelist, by = "hugoGeneSymbol",
    molecularProfileIds = "brca_tcga_rna_seq_v2_mrna"
  )
  # Extract the RNA-Seq data from the MultiAssayExperiment
  tcgaRseqGenelist <- assay(brca_rnaseq[["brca_tcga_rna_seq_v2_mrna"]])
}

```

```

# Function tcgaRseqGenelistERpos #####
#
#
# The function tcgaRseqGenelistERpos obtains RNA-Seq data of a provided genelist
# from TCGA using the cBioPortal access tools
# and delivers only the data of ERpos BRCA samples.
#
# We apply the cBioPortalData package to access data from the cBio Portal
# at www.cbioportal.org
# This will allow to download RNA-Seq data from the TCGA-BRCA cohort.

library(cBioPortalData)
# First we setup some parameters for the cBioportal-access
# Define api
cbio <- cBioPortal()

# Function definition:
# (genelist is a vector of gene symbols)

tcgaRseqGenelistERpos <- function (genelist) {
  # Download BRCA RNA-Seq data for this genelist from cBioPortal
  # as a "MultiAssayExperiment" brca_rnaseq
  brca_rnaseq <- cBioPortalData(
    api = cbio,
    studyId = "brca_tcga",
    genes = genelist, by = "hugoGeneSymbol",
    molecularProfileIds = "brca_tcga_rna_seq_v2_mrna"
  )
  # Extract the RNA-Seq data from the MultiAssayExperiment
  tcgaRseqGenelist <- assay(brca_rnaseq[["brca_tcga_rna_seq_v2_mrna"]])

  # Extract the phenotype data for TCGA-samples (by patientId)
  pheno <- colData(brca_rnaseq)

  # Extract the link-information between
  # the patientId ("primary") and the RNA-seq-colnames ("colname")
  # from the MultiAssayExperiment
  sample_info <- unique(sampleMap(brca_rnaseq)[,2:3])

  # Now we use dplyr functions from tidyR to join
  # the "ER_STATUS_BY_IHC" from pheno
  # with the "colname" from sample_info
  # by linking the cases using the patientId == primary

  pdata <- as.data.frame(pheno) %>%
    dplyr::select(patientId, ER_STATUS_BY_IHC) %>%
    left_join(as.data.frame(sample_info), by = join_by(patientId == primary))

  # We can now use pdata to select only ER-positive samples

  pdata.erpos <- pdata %>% filter(ER_STATUS_BY_IHC == "Positive")

  tcgaRseqGenelistERpos <- tcgaRseqGenelist[, colnames(tcgaRseqGenelist)

```

```
} %in% pdata.erpos$colname]
```

```
# DATA IMPORT ####
```

```
library(dplyr)
```

*Import list of genes from HTG-panel*

```
htgprobes <- pull(read.table("HTG-OncBiomarkerPanel_n2559-Genelist.txt",  
                             header=FALSE, sep=","))
```

*Definition of genelist for Oncotype Recurrence Score*

```
library(genefu)
```

```
# Load the Oncotype gene info table from the genefu package
```

```
data(sig.oncotypedx)
```

```
# Save original copy of this gene info table
```

```
sig.oncotypedx.ori <- sig.oncotypedx
```

```
# CTSL2 (EntrezGene.ID 1515) has been renamed to CTSV in current HUGO versions
```

```
# Thus we need to rename this gene in the new version "sig.oncotypedx.new":
```

```
sig.oncotypedx.new <- sig.oncotypedx
```

```
sig.oncotypedx.new[rownames(sig.oncotypedx.new)=="CTSL2", ]$symbol <- "CTSV"
```

```
rownames(sig.oncotypedx.new) <- sig.oncotypedx.new$symbol
```

```
# Remove the row with the gene "GRB7" unavailable in HTG-panel:
```

```
sig.oncotypedx.htg <- sig.oncotypedx.new[!(rownames(sig.oncotypedx.ori)  
                                           %in% c("GRB7")), ]
```

*Import GGI genelist from genefu package*

```
library(genefu)
```

```
data(sig.ggi)
```

*Import SET-ER/PR index genelist from PMID 31231679 (Sinn 2019) Suppl.Table 2*

```
sig.set18 <- pull(read.table("SET-ERPR-genes_PMDID_31231679.txt",  
                             header=FALSE, sep=","))
```

*Import 355 Penelope signature infos*

```
sig.pen355 <- read.table("Penelope_n355genes_info.txt",  
                        header=TRUE, sep="\t")
```

```

# ANALYSIS ####

# Recurrence Score ####

# IMPORT TCGA-RNAseq data for oncotype genes

genelist <- sig.oncotypedx.new$symbol

# RNAseq for all TCGA-BRCA (including ERneg)
tcga.RS.Rseq <- tcgaRseqGenelist(genelist)

## harmonizing input:
## removing 8 colData rownames not in sampleMap 'primary'

# RNAseq for ERpos TCGA-BRCA
tcga.RS.Rseq.ERpos <- tcgaRseqGenelistERpos(genelist)

## harmonizing input:
## removing 8 colData rownames not in sampleMap 'primary'

# *****
# Calculate Oncotype Recurrence Score for ERpos TCGA samples *
# *****

# Format data for oncotype function from genefu package
#
# Transpose the count matrix of the ERpos subset
tcga <- t(tcga.RS.Rseq.ERpos)

# Log2 transform count data
log2tcga <- log2(tcga + 1)

# First we run the oncotypedxCTSV function for oncotype classification
# This includes the GRB7 gene and gene symbol "CTSL2" is updated to "CTSV".

tcga.onc.ctsv <- oncotypedxCTSV(data=log2tcga,
                               annot=sig.oncotypedx.new,
                               do.mapping = TRUE, verbose = TRUE)

# Next we run the oncotypedxHTG function for oncotype classification
# This excludes the GRB7 gene (but includes "CTSL2" updated to "CTSV").

tcga.onc.htg <- oncotypedxHTG(data=log2tcga,
                              annot=sig.oncotypedx.new,
                              do.mapping = TRUE, verbose = TRUE)

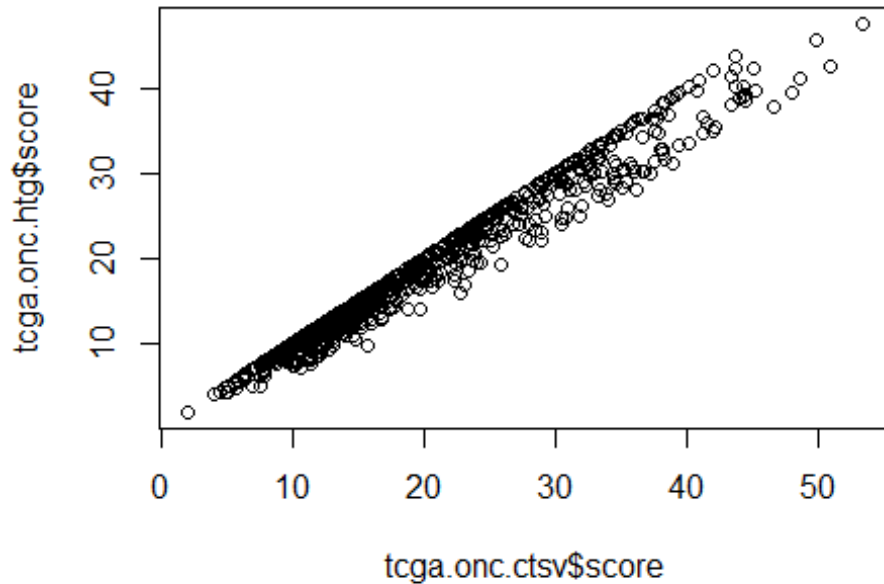
# *****

```

```
# COMPARE OBTAINED RESULTS FOR RS #####
```

```
# Finally we compare the results obtained from the two different functions
```

```
plot(tcga.onc.ctsv$score, tcga.onc.htg$score)
```



```
cor(tcga.onc.ctsv$score, tcga.onc.htg$score)
```

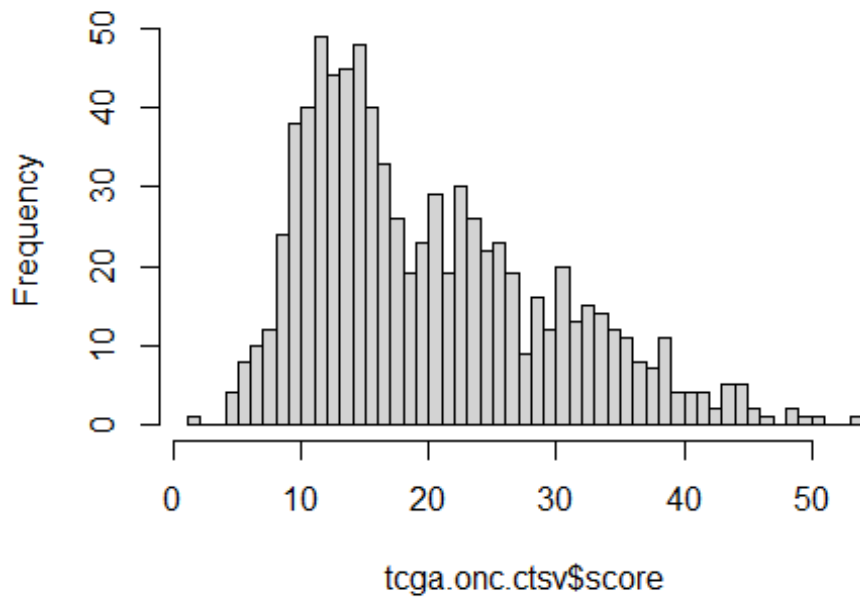
```
## [1] 0.9839576
```

```
# Interpretation: Some minor differences when GRB7 is omitted (oncotypedxHTG),  
# but still very good consistency with R=0.98
```

```
hist(tcga.onc.ctsv$score, breaks = 40)
```

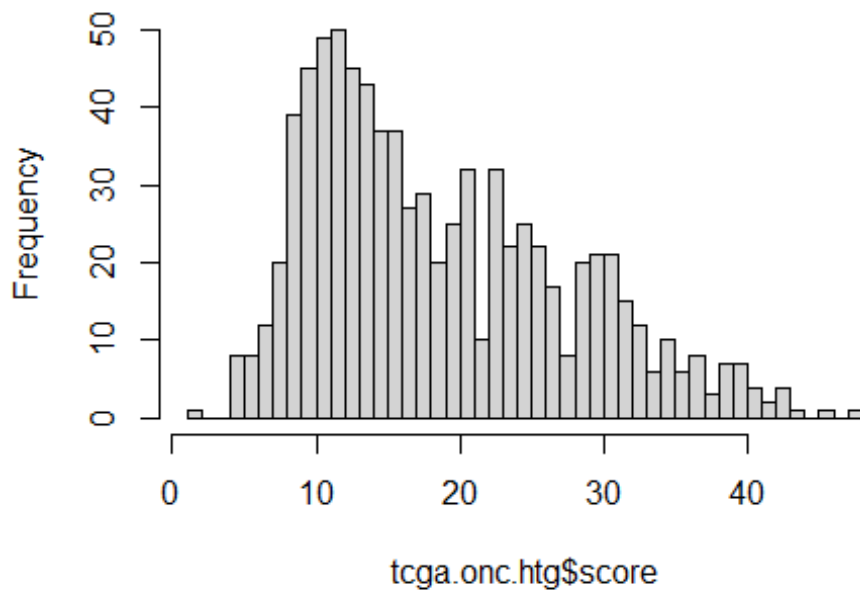


**Histogram of tcga.onc.ctsv\$score**



```
hist(tcga.onc.htg$score, breaks = 40)
```

**Histogram of tcga.onc.htg\$score**



```
# Distributions of oncotype recurrences scores  
# match published data (PMID 32565552)
```

```
table(tcga.onc.ctsv$risk)
```

```
##
##  0 0.5  1
## 137 476 199

table(tcga.onc.htg$risk)

##
##  0 0.5  1
## 182 456 174

# risk groups similar

# *****
```

```

# Genomic Grade Index #####

# *****
# Calculate GGI for TCGA samples *
# *****
#

# # Import GGI genelist from genefu package
# # Has already been performed in DATA IMPORT section above !
# library(genefu)
# data(sig.ggi)

# count number of GGI genes with HUGO.gene.symbol
sum(!is.na(sig.ggi$HUGO.gene.symbol))

## [1] 112

# 112 genes

# count number of GGI genes in HTG-panel
sum(sig.ggi$HUGO.gene.symbol %in% htgprobes)

## [1] 56

# 56 genes available

# Use HUGO.gene.symbol as probe name for TCGA mapping
library(dplyr)
library(tibble)
HUGO.sig.ggi <- sig.ggi %>%
  filter(!is.na(HUGO.gene.symbol)) %>% # keep only probes with HUGO gene symbol
  mutate(probe = HUGO.gene.symbol) %>% # replace Affy probes with gene symbol
  distinct(probe, .keep_all = TRUE) %>% # remove duplicates
  mutate(weight = grade-2) %>% # weights: +1 / -1
  dplyr::select(probe, weight, HUGO.gene.symbol,
                EntrezGene.ID) %>% # select cols
  remove_rownames() %>% # remove Affy-rownames
  arrange(., probe) # order rows by gene names

# Get TCGA RNA-seq data of the breast cancer cohort for
# the genes from GGI gene list:

# Define genelist
genelist <- HUGO.sig.ggi$probe

# RNAseq for all TCGA-BRCA (including ERneg)
tcga.ggi.Rseq <- tcgaRseqGenelist(genelist)

## harmonizing input:
## removing 8 colData rownames not in sampleMap 'primary'

# RNAseq for ERpos TCGA-BRCA
tcga.ggi.Rseq.ERpos <- tcgaRseqGenelistERpos(genelist)

```

```
## harmonizing input:
## removing 8 colData rownames not in sampleMap 'primary'
```

*Calculate GGI all TCGA-BRCA (including ERneg)*

```
# Transpose the count matrix of ALL TCGA samples
```

```
tcga <- t(tcga.ggi.Rseq)
```

```
# Log2 transform count data
```

```
log2tcga <- log2(tcga + 1)
```

```
# order columns by gene names
```

```
log2tcga <- log2tcga[,order(colnames(log2tcga))]
```

```
# *** Use all 85 ggi-genes available in TCGA dataset: ***
```

```
tcga.sig.ggi <- HUGO.sig.ggi %>%  
  filter(probe %in% colnames(log2tcga))
```

```
# check identity:
```

```
stopifnot(colnames(log2tcga) == tcga.sig.ggi$probe)
```

```
# Compute GGI for TCGA
```

```
ggi.score <- rep(NA,nrow(log2tcga)) # empty vector for results
```

```
names(ggi.score) <- rownames(log2tcga)
```

```
for (i in 1:nrow(log2tcga)){
```

```
  ggi.score[i] <- mean(log2tcga[i,]*tcga.sig.ggi$weight)
```

```
}
```

```
# *** Use only 46 ggi-genes available in HTG and TCGA dataset: ***
```

```
tcga.sig.ggi.htg <- HUGO.sig.ggi %>%
```

```
  filter(probe %in% colnames(log2tcga)) %>%
```

```
  filter(probe %in% htgprobes)
```

```
# TCGA RNA-Seq or subset of 46 HTG ggi-genes
```

```
log2tcga.htg <- log2tcga[, colnames(log2tcga) %in% htgprobes]
```

```
# check identity:
```

```
stopifnot(colnames(log2tcga.htg) == tcga.sig.ggi.htg$probe)
```

```
# Compute GGI for TCGA using only HTG genes
```

```
ggi.score.htg <- rep(NA,nrow(log2tcga.htg)) # empty vector for results
```

```
names(ggi.score.htg) <- rownames(log2tcga.htg)
```

```
for (i in 1:nrow(log2tcga.htg)){
```

```
  ggi.score.htg[i] <- mean(log2tcga.htg[i,]*tcga.sig.ggi.htg$weight)
```

```
}
```

*Calculate GGI for TCGA-ERpos-Only*

```
# Transpose the count matrix of ERpos TCGA samples
```

```
tcga <- t(tcga.ggi.Rseq.ERpos)
```

```

# log2 transform count data
log2tcga <- log2(tcga + 1)

# order columns by gene names
log2tcga <- log2tcga[,order(colnames(log2tcga))]

# *** Use only 46 ggi-genes available in HTG and TCGA dataset: ***

tcga.sig.ggi.htg <- HUGO.sig.ggi %>%
  filter(probe %in% colnames(log2tcga)) %>%
  filter(probe %in% htgprobes)

# TCGA RNA-Seq or subset of 46 HTG ggi-genes
log2tcga.htg <- log2tcga[, colnames(log2tcga) %in% htgprobes]

# check identity:
stopifnot(colnames(log2tcga.htg) == tcga.sig.ggi.htg$probe)

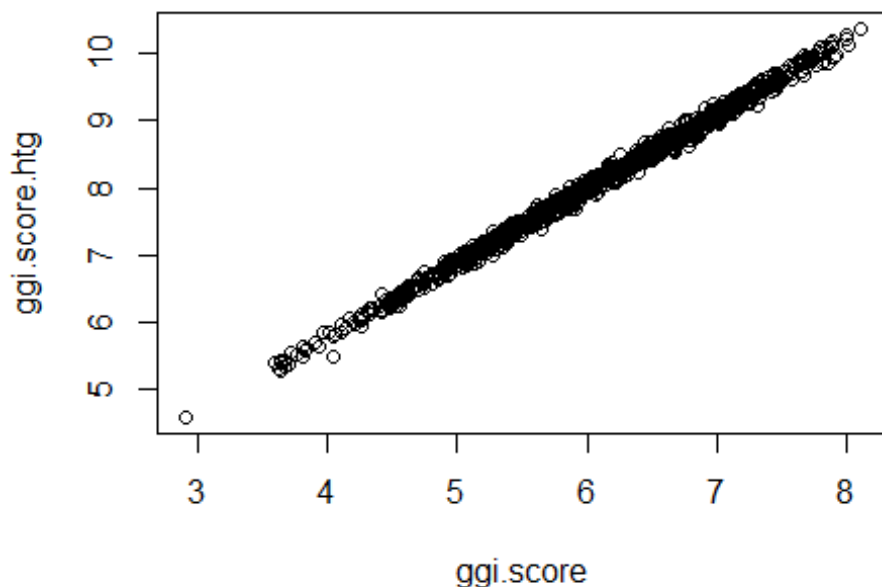
# Compute GGI for TCGA using only HTG genes
ggi.score.htg.ERpos <- rep(NA,nrow(log2tcga.htg)) # empty vector for results
names(ggi.score.htg.ERpos) <- rownames(log2tcga.htg)
for (i in 1:nrow(log2tcga.htg)){
  ggi.score.htg.ERpos[i] <- mean(log2tcga.htg[i,]*tcga.sig.ggi.htg$weight)
}

# COMPARE OBTAINED RESULTS FOR GGI #####

# *** ggi from 85 total genes and 46 htg genes in ALL TCGA BRCA: ***

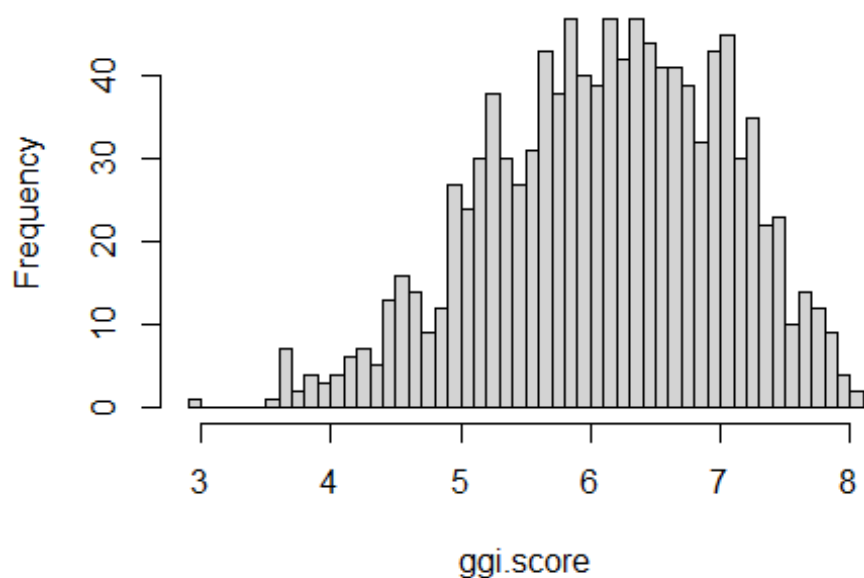
plot(ggi.score, ggi.score.htg)

```



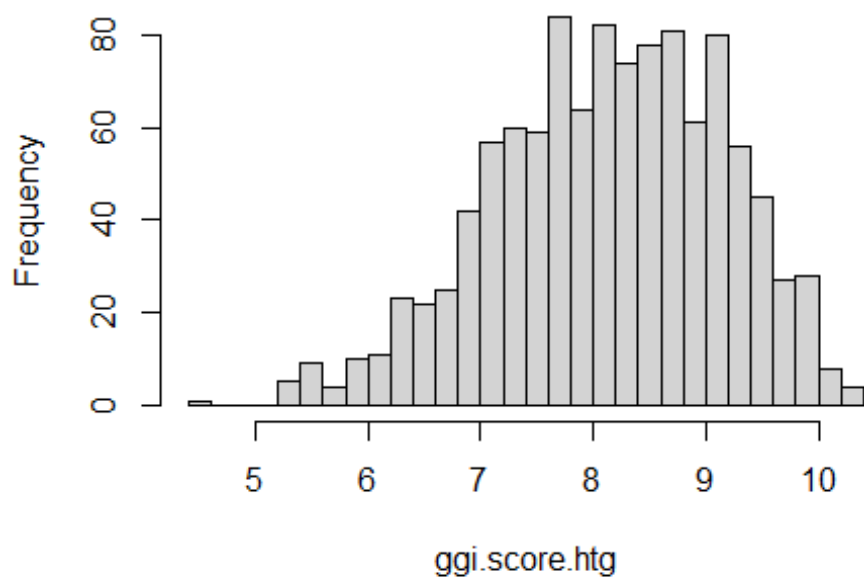
```
cor(ggi.score, ggi.score.htg)
## [1] 0.9974926
# Interpretation: nearly identical, R=0,997
hist(ggi.score, breaks = 40)
```

**Histogram of ggi.score**



```
hist(ggi.score.htg, breaks = 40)
```

**Histogram of ggi.score.htg**





```

# SET Index #####

# *****
# Calculate SET-ER/PR Index for TCGA samples *
# *****
#

# # Import SET-ER/PR index genelist from PMID 31231679 (Sinn 2019) Suppl.Table 2
# # Has already been performed in DATA IMPORT section above !
# sig.set18 <- pull(read.table("SET-ERPR-genes_PMIID 31231679.txt",
#                               header=FALSE, sep=","))

sig.set18.htg <- sig.set18[sig.set18 %in% htgprobes]
# 12 genes available

# SET18 genes missing:
sig.set18[!(sig.set18 %in% htgprobes)]

## [1] "NPY1R" "AZGP1" "ABAT" "ADCY1" "MRPS30" "KCNE4"

# "NPY1R" "AZGP1" "ABAT" "ADCY1" "MRPS30" "KCNE4"

##### #
# Get TCGA RNA-seq data of the breast cancer cohort for
# the genes from SET-ER/PR index gene list:

# Define genelist
genelist <- sig.set18

tcga.set18.Rseq <- tcgaRseqGenelist(genelist) # including ERneg

## harmonizing input:
## removing 8 colData rownames not in sampleMap 'primary'

tcga.set18.Rseq.ERpos <- tcgaRseqGenelistERpos(genelist)

## harmonizing input:
## removing 8 colData rownames not in sampleMap 'primary'

SET index for ALL TCGA-BRCA (including ERneg)

# Transpose the count matrix of ALL TCGA samples
tcga <- t(tcga.set18.Rseq)

# Log2 transform count data
log2tcga <- log2(tcga + 1)

# order columns by gene names
log2tcga <- log2tcga[,order(colnames(log2tcga))]

# *** Use all 18 SET-ER/PR index genes available in TCGA dataset: ***

```



```

# Compute mean of set18 for TCGA
set.score <- rep(NA,nrow(log2tcga)) # empty vector for results
names(set.score) <- rownames(log2tcga)
for (i in 1:nrow(log2tcga)){
  set.score[i] <- mean(log2tcga[i,])
}

# *** Use only the 12 SET-ER/PR-index genes available in HTG panel: ***

# TCGA RNA-Seq or subset of 46 HTG ggi-genes
log2tcga.htg <- log2tcga[, colnames(log2tcga) %in% sig.set18.htg]

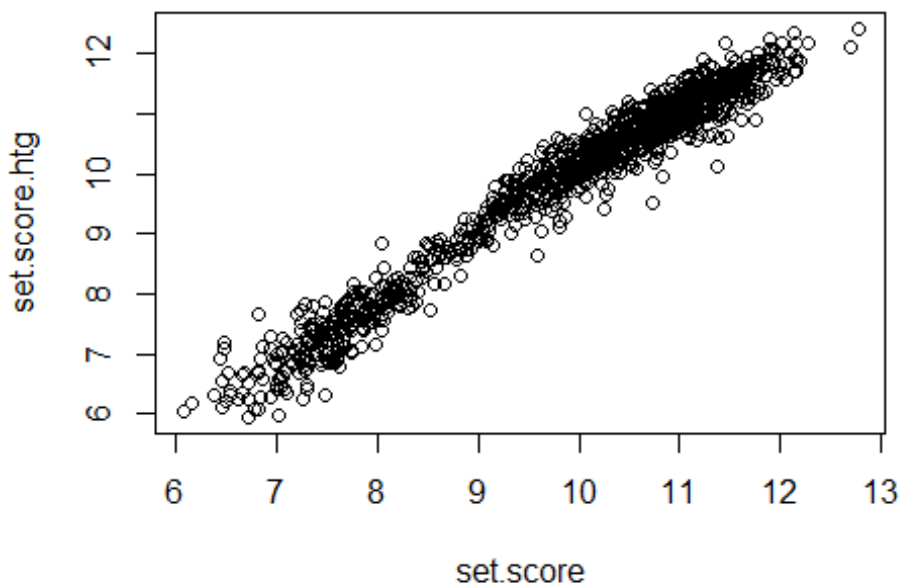
# Compute mean of set18.htg for TCGA using only HTG genes
set.score.htg <- rep(NA,nrow(log2tcga.htg)) # empty vector for results
names(set.score.htg) <- rownames(log2tcga.htg)
for (i in 1:nrow(log2tcga.htg)){
  set.score.htg[i] <- mean(log2tcga.htg[i,])
}

# COMPARE OBTAINED RESULTS FOR SET18 ####

# *** SET18 from 18 total genes and 12 htg genes in TCGA dataset: ***
# (including ERneg)

plot(set.score, set.score.htg)

```



```

cor(set.score, set.score.htg)

## [1] 0.9795561

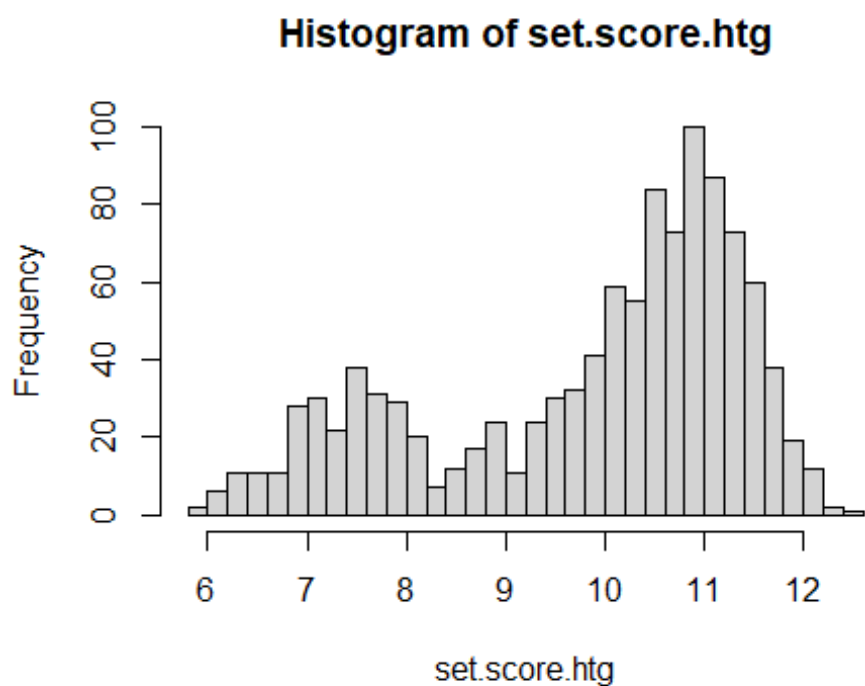
```

```
# Interpretation: nearly identical,  $R=0.98$ 
```

```
hist(set.score, breaks = 40)
```



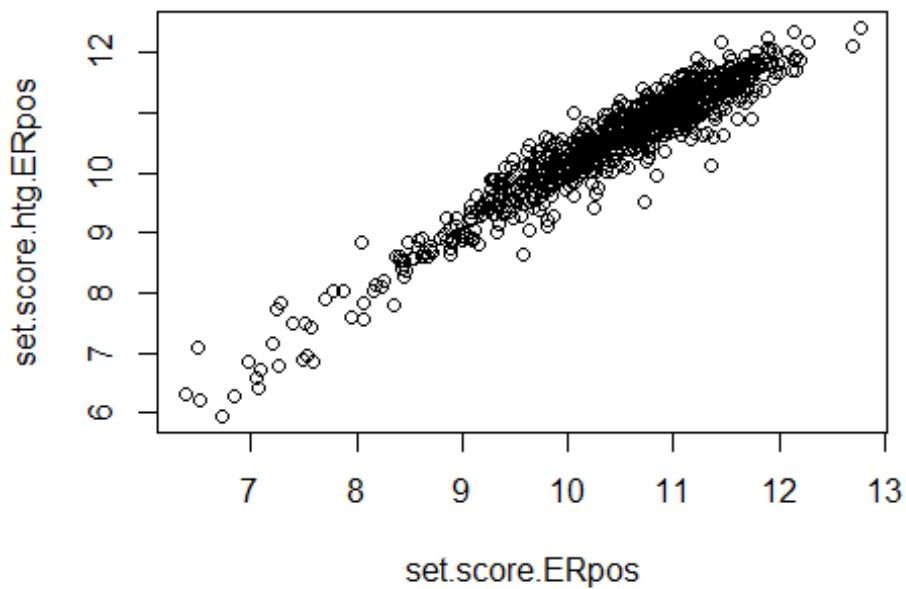
```
hist(set.score.htg, breaks = 40)
```



```
# Plot data only for ERpos-BRCA
```

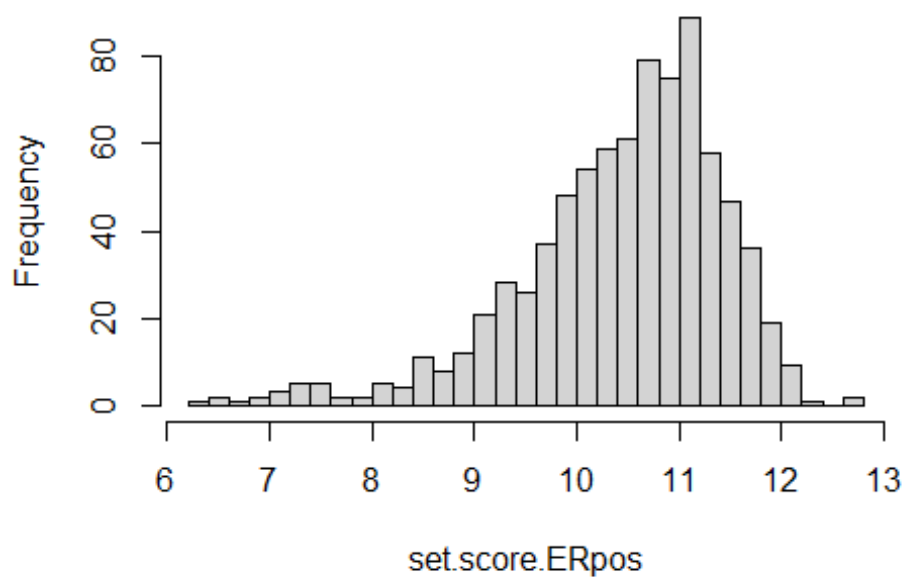
```
set.score.ERpos <- set.score[names(set.score)
```

```
%in% colnames(tcga.set18.Rseq.ERpos)]  
  
set.score.htg.ERpos <- set.score.htg[names(set.score.htg)  
%in% colnames(tcga.set18.Rseq.ERpos)]  
  
plot(set.score.ERpos, set.score.htg.ERpos)
```



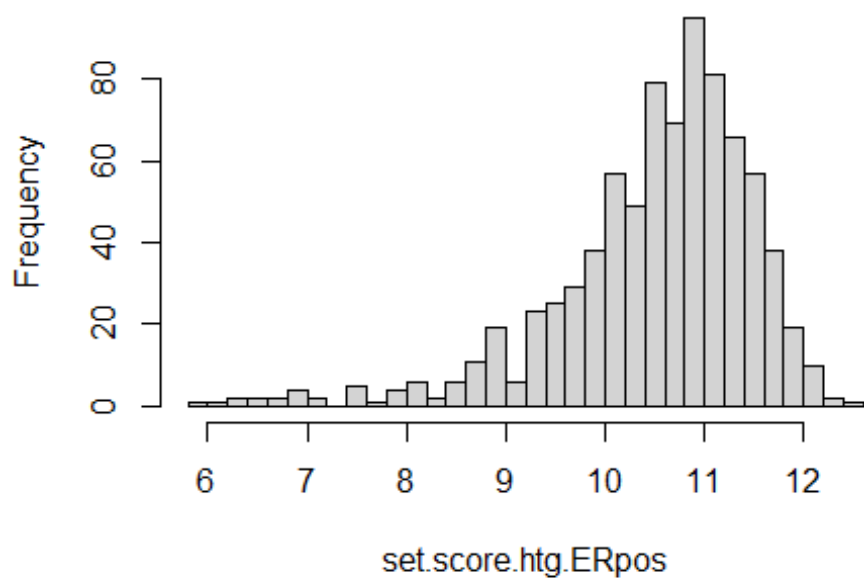
```
cor(set.score.ERpos, set.score.htg.ERpos)  
## [1] 0.9556287  
  
# Interpretation: strong correlation, R=0.956  
  
hist(set.score.ERpos, breaks = 40)
```

**Histogram of set.score.ERpos**



```
hist(set.score.htg.ERpos, breaks = 40)
```

**Histogram of set.score.htg.ERpos**



```

# PENELOPE-Signature #####

# *****
# Define subsets of 355 Penelope signature *
# *****
#

# # Import 355 Penelope signature infos
# # Has already been performed in DATA IMPORT section above !
# sig.pen355 <- read.table("Penelope_n355genes_info.txt",
#                          header=TRUE, sep="\t")
#

# Use TCGA RNA-Seq data as dataset:

# Get TCGA RNA-seq data of the breast cancer cohort for
# the genes from SET-ER/PR index gene list:

# Define genelist
genelist <- sig.pen355$Gene

tcga.pen355.Rseq <- tcgaRseqGenelist(genelist)

## harmonizing input:
## removing 8 colData rownames not in sampleMap 'primary'

tcga.pen355.Rseq.ERpos <- tcgaRseqGenelistERpos(genelist)

## harmonizing input:
## removing 8 colData rownames not in sampleMap 'primary'

# Only 323 of the 355 genes are available in the TCGA dataset from cBioPortal

sig.pen323 <- sig.pen355[sig.pen355$Gene %in% rownames(tcga.pen355.Rseq.ERpos),]

# re-order the genes in rows of RNA-Seq matrix according to sig.pen.323
tcga.pen355.Rseq.ERpos <- tcga.pen355.Rseq.ERpos[sig.pen323$Gene,]

# calculate signature scores
stopifnot(sig.pen323$Gene == rownames(tcga.pen355.Rseq.ERpos))

# Transpose the count matrix of ALL TCGA samples
tcga <- t(tcga.pen355.Rseq.ERpos)

# log2 transform count data
log2tcga <- log2(tcga + 1)

#### Calculate the subcluster mean values for all samples in dataset:

# Define Subcluster signatures to calculate:

```

```

pen322.subclu <- unique(sig.pen323$SubCluster_1080pairedSamples)

# Define RNAseq data:
rseqdata <- log2tcga

# define dataframe for results

rseqdata.pen322.subclu.scores <- data.frame(matrix(NA,
                                                    nrow = nrow(rseqdata),
                                                    ncol = length(pen322.subclu)))
rownames(rseqdata.pen322.subclu.scores) <- rownames(rseqdata)
colnames(rseqdata.pen322.subclu.scores) <- pen322.subclu

# Loop over all pen322 subcluster with second inner loop to calculate
# mean-score of subclusters for all samples in rseqdata

for (i in 1:length(pen322.subclu)) { # Loop over pen322 subclusters
  genes <- sig.pen323$Gene[sig.pen323$SubCluster_1080pairedSamples
                           %in% pen322.subclu[i]] # select genes of subcluster
  for (k in 1:nrow(rseqdata)) { # Loop over samples to calculate score
    rseqdata.pen322.subclu.scores[k,i] <- mean(rseqdata[k, genes], na.rm = TRUE)
  }
}

# Save re-named result:

tcgaERpos.pen322.subclu.scores <- rseqdata.pen322.subclu.scores

##### #

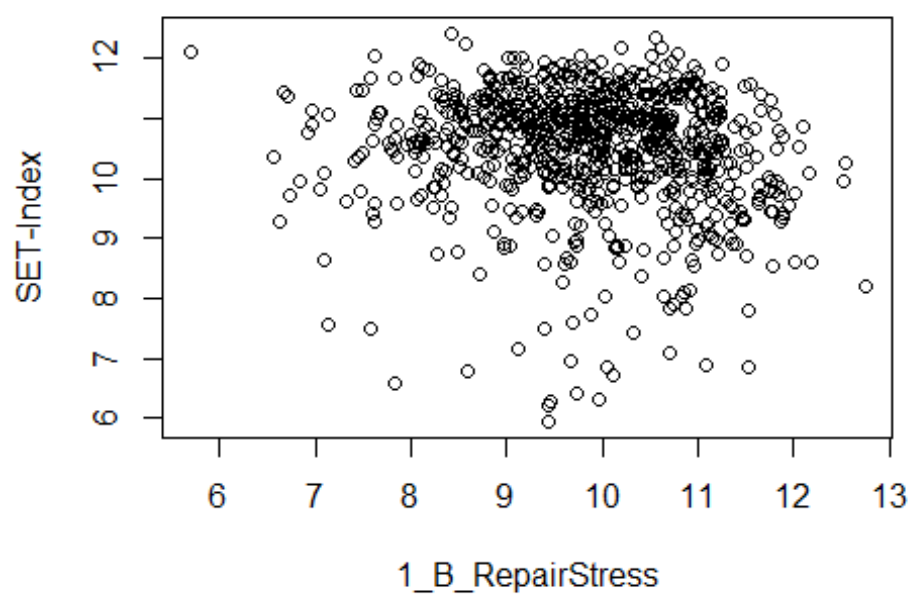
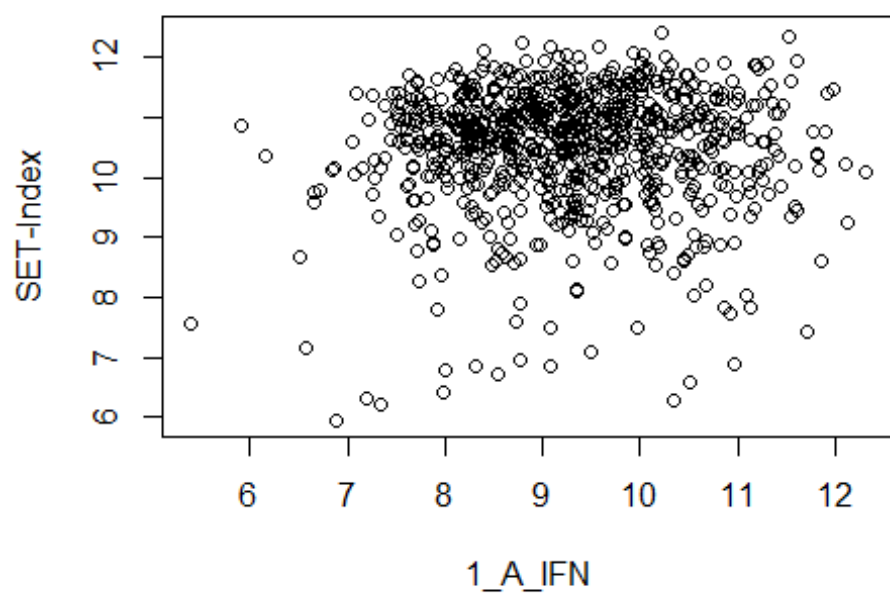
```

```
# Compare PENELOPE-Subcluster-Signatures to other GeneSignatures ####
```

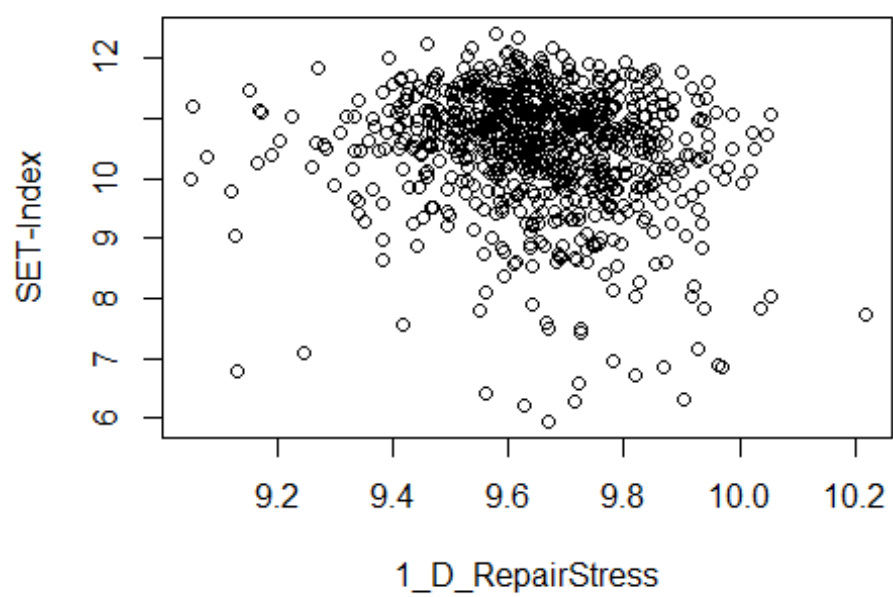
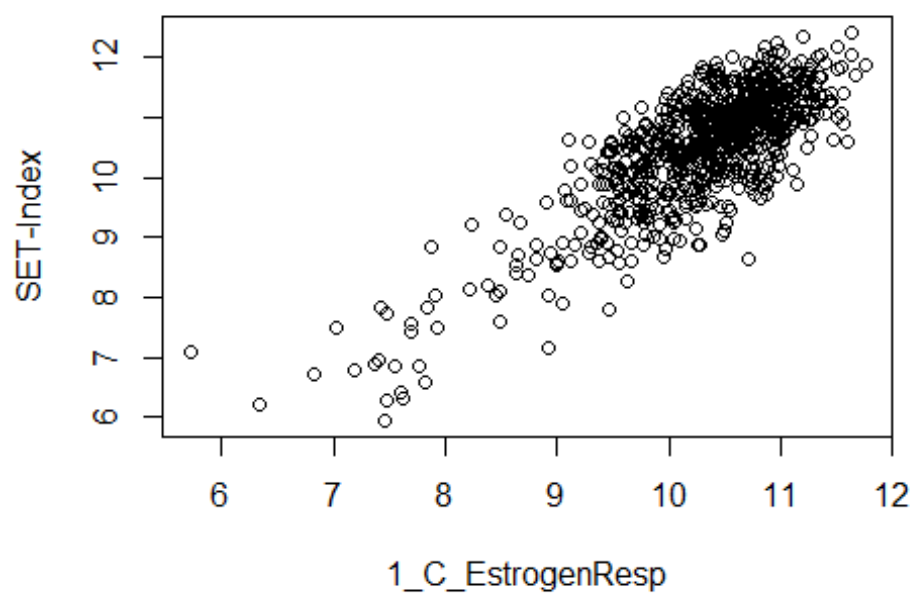
```
# SET-Index:
```

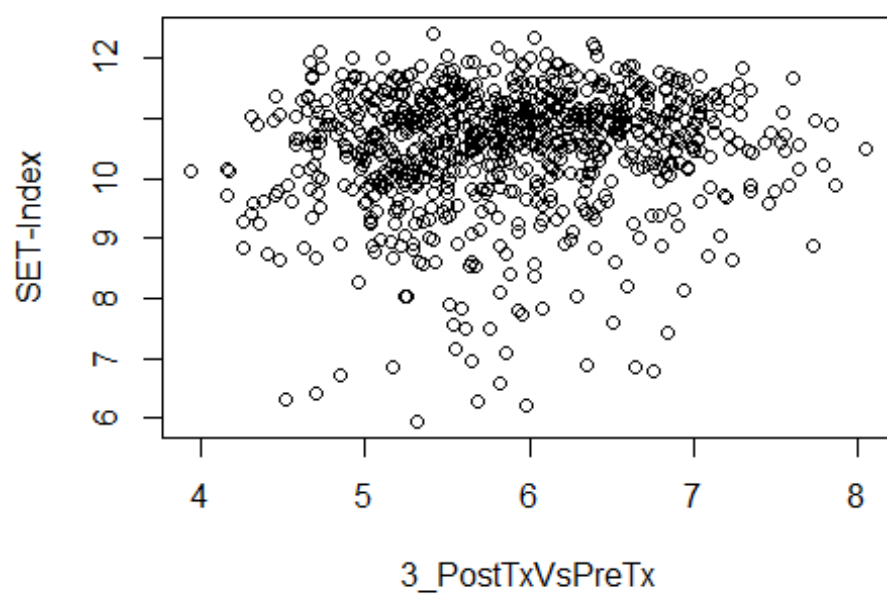
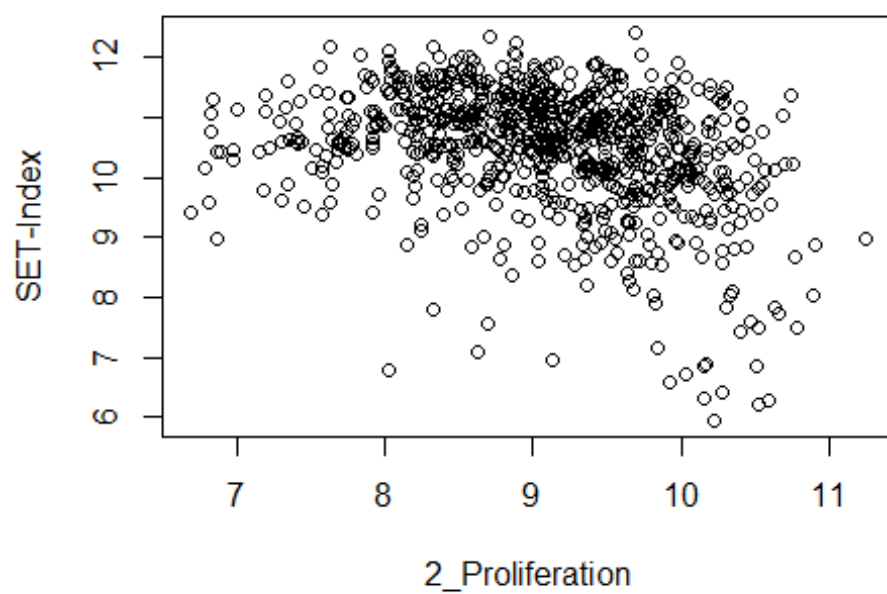
```
stopifnot(rownames(tcgaERpos.pen322.subclu.scores) == names(set.score.htg.ERpos))
```

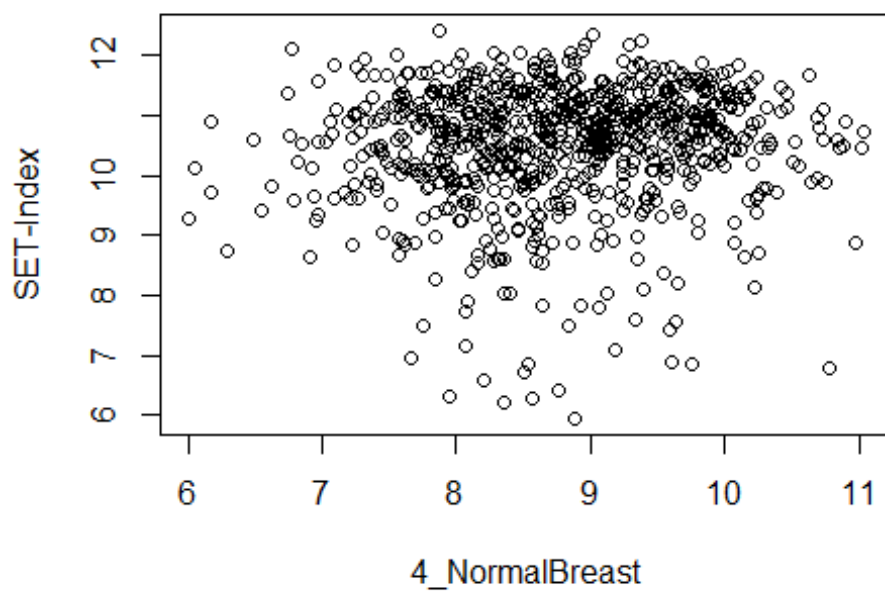
```
for (i in 1:length(pen322.subclu)) {  
  plot(tcgaERpos.pen322.subclu.scores[,i], set.score.htg.ERpos,  
       xlab = pen322.subclu[i], ylab = "SET-Index")  
}
```







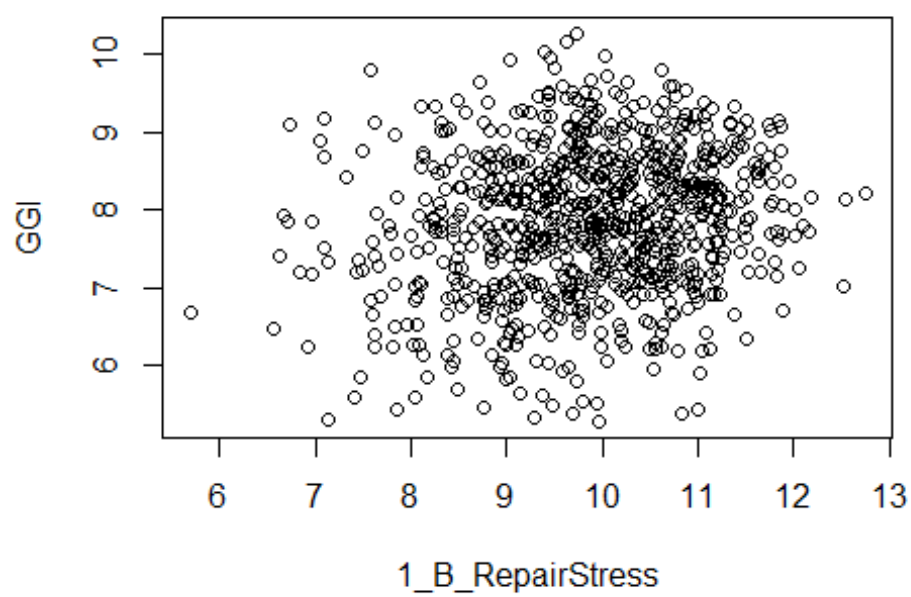
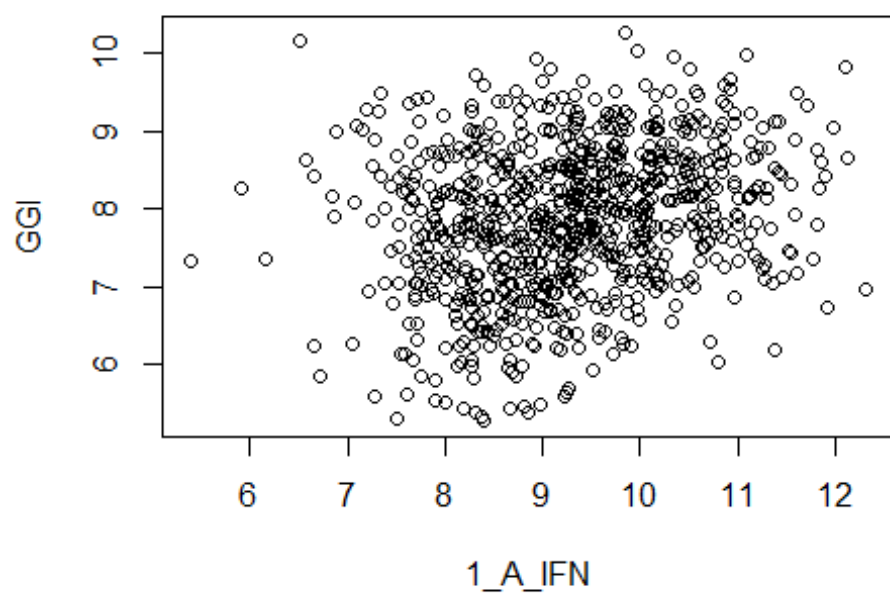


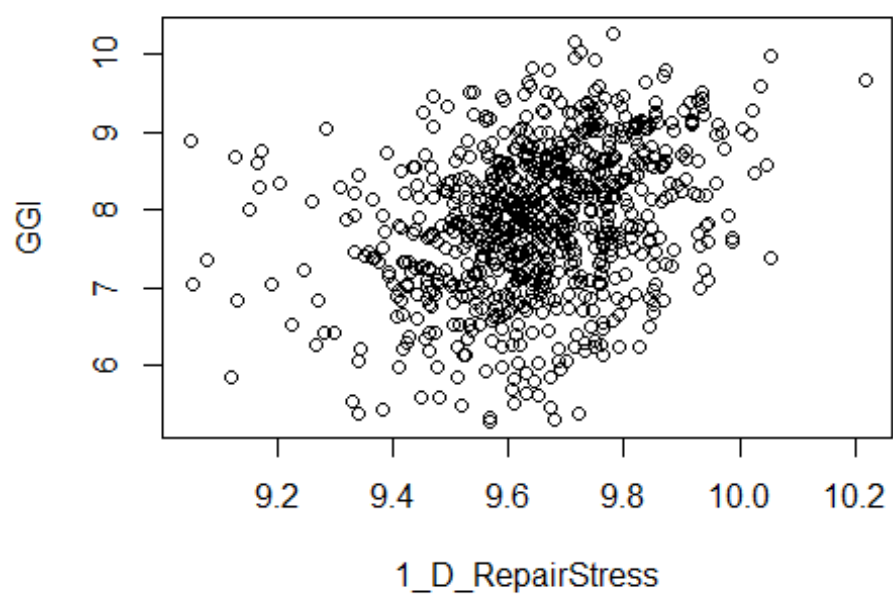
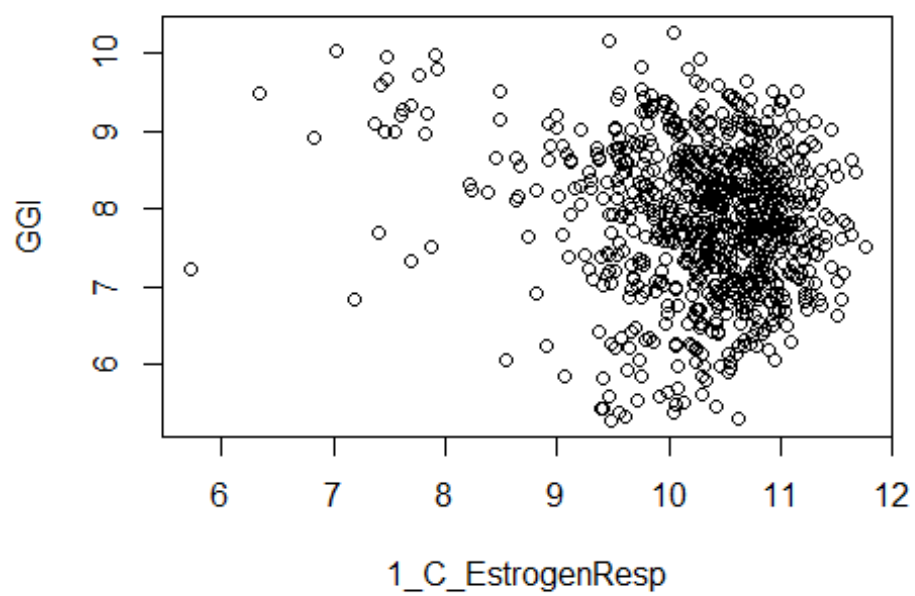


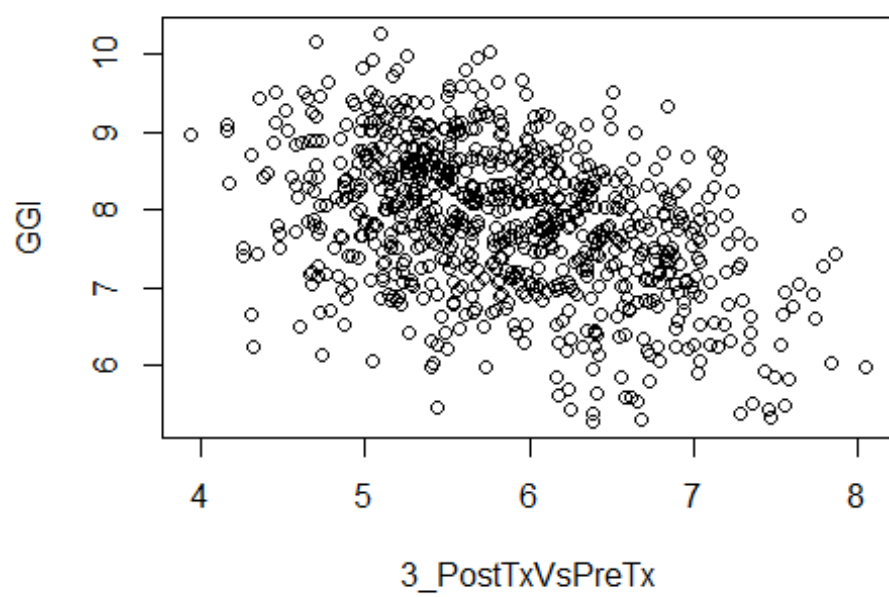
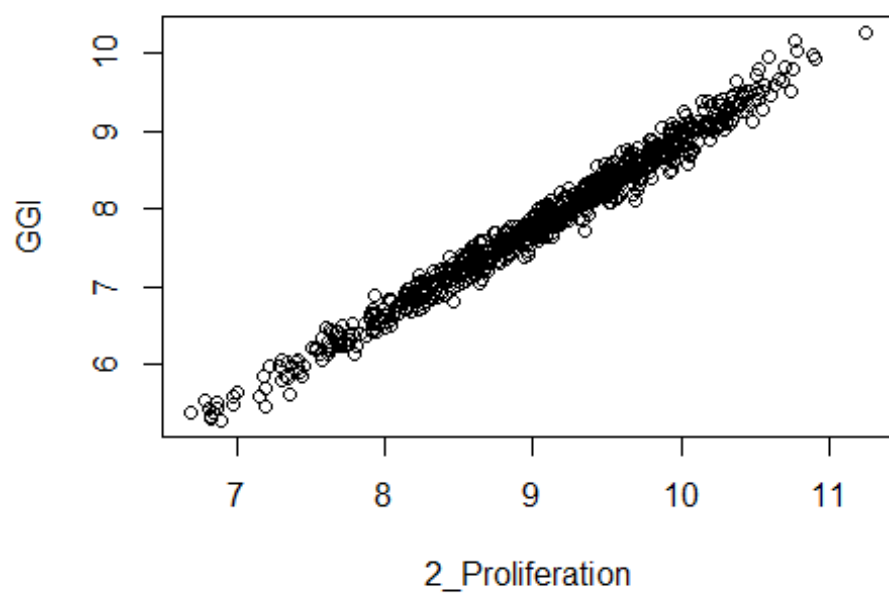
```
# GGI:

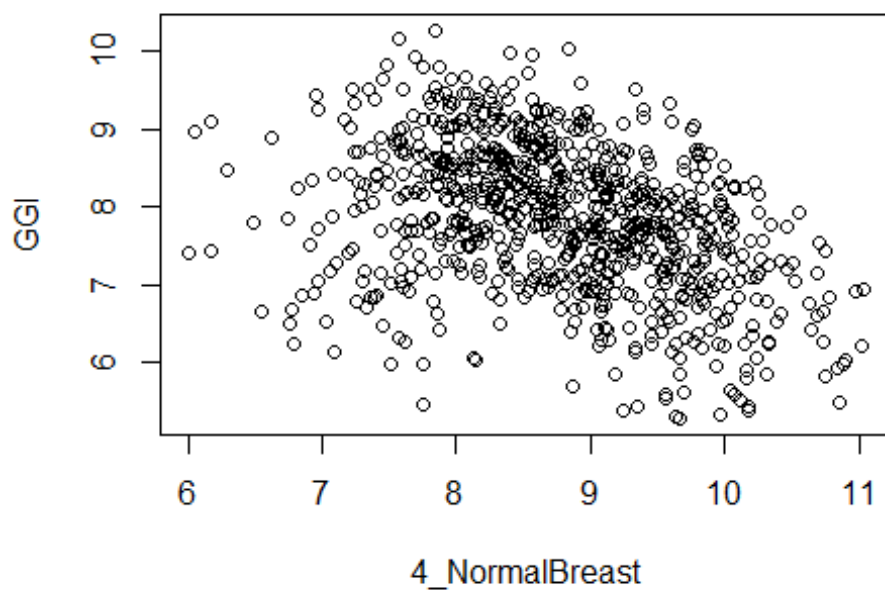
stopifnot(rownames(tcgaERpos.pen322.subclu.scores) == names(ggi.score.htg.ERpos))

for (i in 1:length(pen322.subclu)) {
  plot(tcgaERpos.pen322.subclu.scores[,i], ggi.score.htg.ERpos,
       xlab = pen322.subclu[i], ylab = "GGI")
}
```





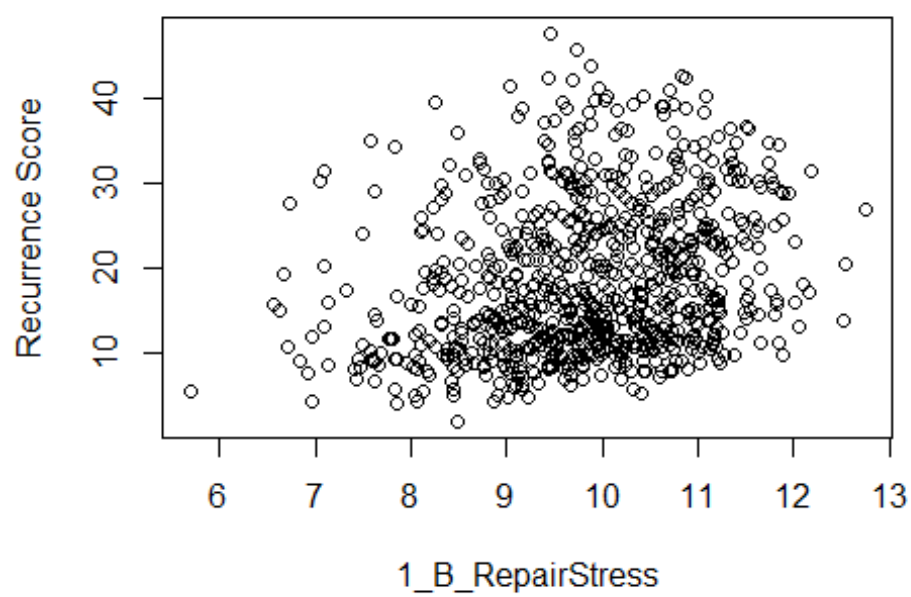
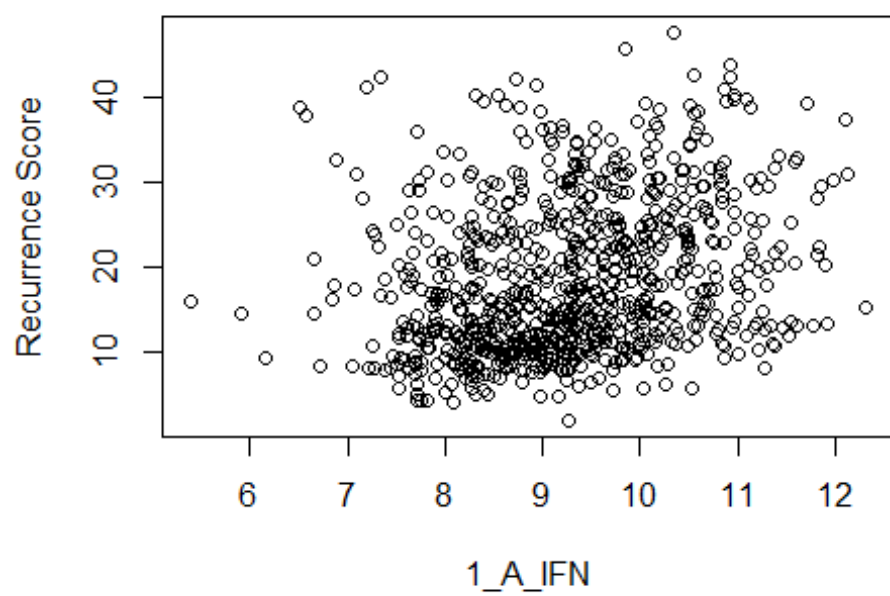




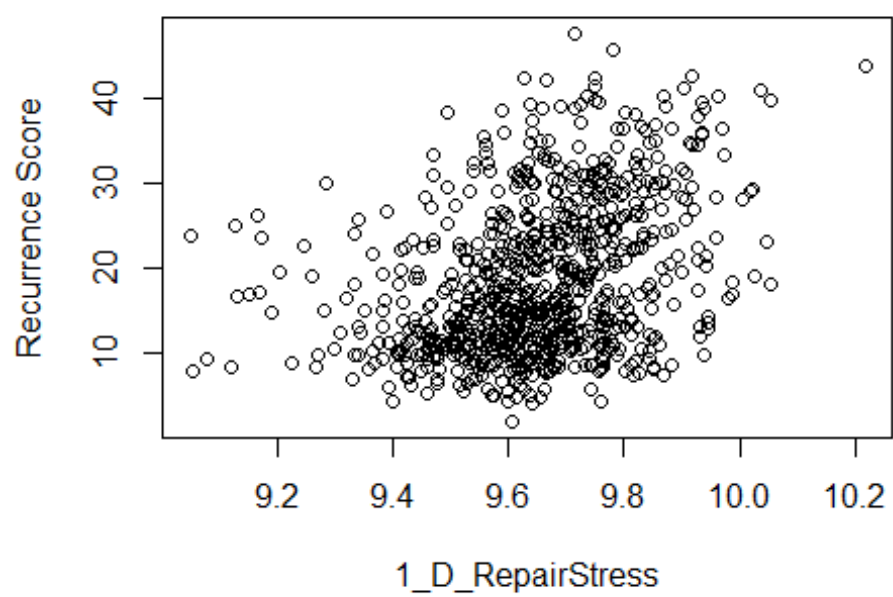
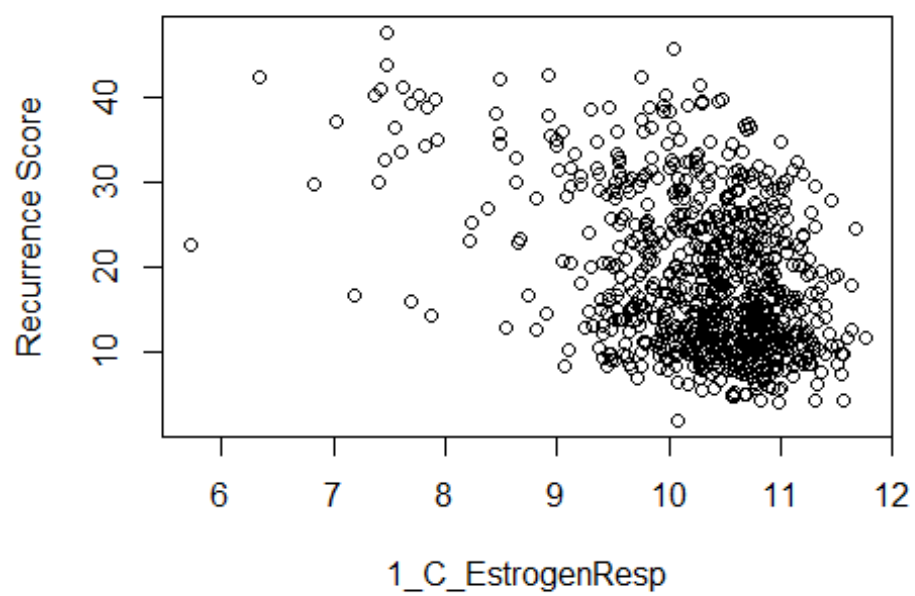
*# Recurrence Score:*

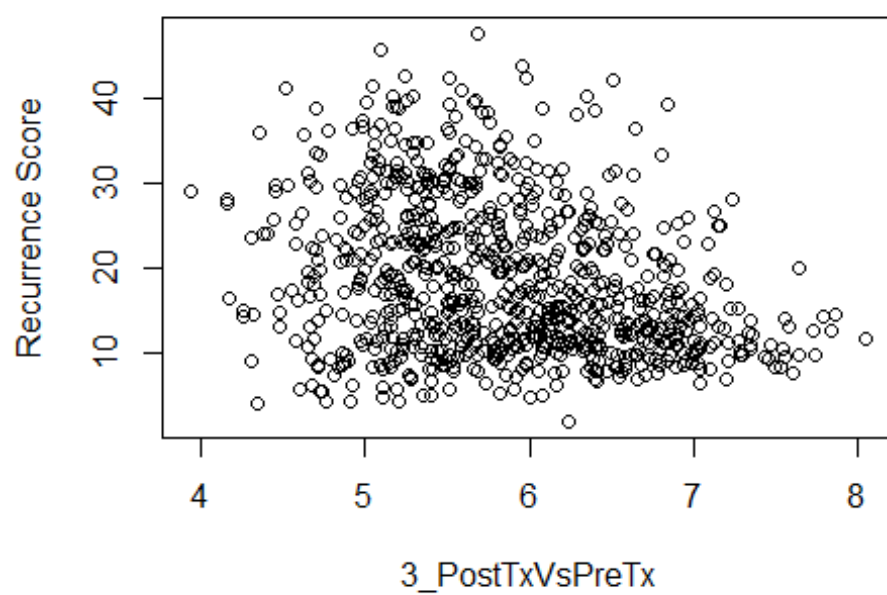
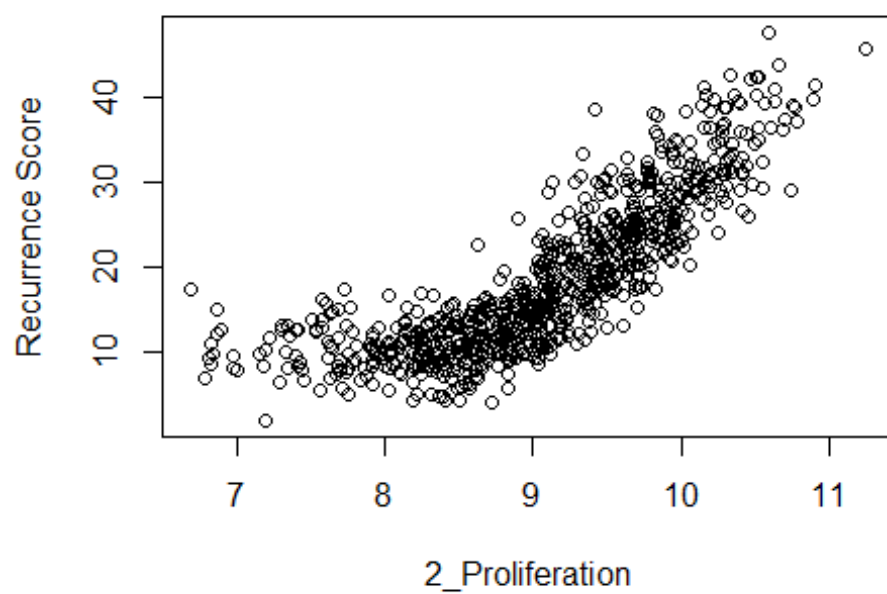
```
stopifnot(rownames(tcgaERpos.pen322.subclu.scores) == names(tcga.onc.htg$score))
```

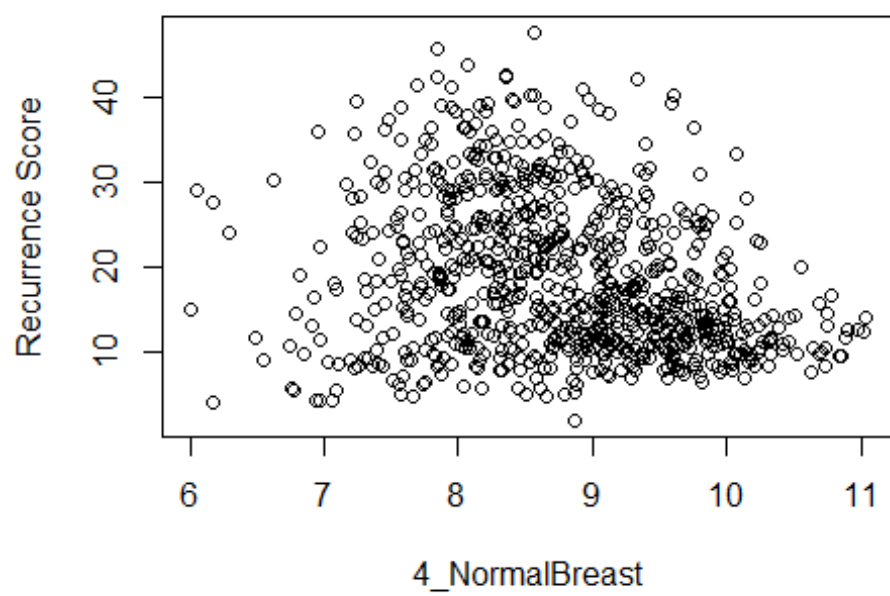
```
for (i in 1:length(pen322.subclu)) {
  plot(tcgaERpos.pen322.subclu.scores[,i], tcga.onc.htg$score,
       xlab = pen322.subclu[i], ylab = "Recurrence Score")
}
```











# # SESSION INFO ####

## sessionInfo()

```
## R version 4.3.0 (2023-04-21 ucrt)
## Platform: x86_64-w64-mingw32/x64 (64-bit)
## Running under: Windows 11 x64 (build 22631)
##
## Matrix products: default
##
## locale:
## [1] LC_COLLATE=German_Germany.utf8  LC_CTYPE=German_Germany.utf8
## [3] LC_MONETARY=German_Germany.utf8 LC_NUMERIC=C
## [5] LC_TIME=German_Germany.utf8
##
## time zone: Europe/Berlin
## tzcode source: internal
##
## attached base packages:
## [1] stats4      stats      graphics  grDevices  utils      datasets  methods
## [8] base
##
## other attached packages:
## [1] tibble_3.2.1           cBioPortalData_2.12.0
## [3] MultiAssayExperiment_1.26.0 SummarizedExperiment_1.30.2
## [5] GenomicRanges_1.52.0   GenomeInfoDb_1.36.1
## [7] IRanges_2.34.1         S4Vectors_0.38.1
## [9] MatrixGenerics_1.12.3  matrixStats_1.0.0
## [11] AnVIL_1.12.3           genefu_2.32.0
## [13] AIMS_1.32.0            Biobase_2.60.0
## [15] BiocGenerics_0.46.0    e1071_1.7-13
## [17] iC10_1.5               iC10TrainingData_1.3.1
## [19] impute_1.74.1          pamr_1.56.1
## [21] cluster_2.1.4          biomaRt_2.56.1
## [23] survcomp_1.50.0        prodlim_2023.03.31
## [25] survival_3.5-5         dplyr_1.1.2
##
## loaded via a namespace (and not attached):
## [1] rstudioapi_0.15.0      jsonlite_1.8.5
## [3] magrittr_2.0.3         GenomicFeatures_1.52.1
## [5] SuppDists_1.1-9.7      rmarkdown_2.22
## [7] BiocIO_1.10.0          zlibbioc_1.46.0
## [9] vctrs_0.6.2            Rsamtools_2.16.0
## [11] memoise_2.0.1          RCurl_1.98-1.12
## [13] htmltools_0.5.5        S4Arrays_1.0.5
## [15] progress_1.2.2         lambda.r_1.2.4
## [17] curl_5.0.2             parallelly_1.36.0
## [19] KernSmooth_2.23-20     htmlwidgets_1.6.2
## [21] futile.options_1.0.1   cachem_1.0.8
## [23] GenomicAlignments_1.36.0 mime_0.12
## [25] lifecycle_1.0.3       pkgconfig_2.0.3
## [27] Matrix_1.6-1           R6_2.5.1
## [29] fastmap_1.1.1          GenomeInfoDbData_1.2.10
## [31] future_1.33.0          shiny_1.7.4.1
## [33] digest_0.6.31          RaggedExperiment_1.24.0
## [35] AnnotationDbi_1.62.2   RSQLite_2.3.1
```

## [37]	filelock_1.0.2	RTCGAToolbox_2.30.0
## [39]	fansi_1.0.4	RJSONIO_1.3-1.8
## [41]	httr_1.4.6	abind_1.4-5
## [43]	compiler_4.3.0	proxy_0.4-27
## [45]	withr_2.5.0	bit64_4.0.5
## [47]	BiocParallel_1.34.2	DBI_1.1.3
## [49]	highr_0.10	lava_1.7.2.1
## [51]	rappdirs_0.3.3	DelayedArray_0.26.7
## [53]	rjson_0.2.21	tools_4.3.0
## [55]	httpuv_1.6.11	future.apply_1.11.0
## [57]	bootstrap_2019.6	glue_1.6.2
## [59]	restfulr_0.0.15	promises_1.2.0.1
## [61]	grid_4.3.0	generics_0.1.3
## [63]	tzdb_0.4.0	class_7.3-21
## [65]	tidyr_1.3.0	data.table_1.14.8
## [67]	hms_1.1.3	xml2_1.3.5
## [69]	utf8_1.2.3	XVector_0.40.0
## [71]	pillar_1.9.0	stringr_1.5.0
## [73]	RCircos_1.2.2	limma_3.56.2
## [75]	later_1.3.1	splines_4.3.0
## [77]	BiocFileCache_2.8.0	lattice_0.21-8
## [79]	rtracklayer_1.60.0	bit_4.0.5
## [81]	tidyselect_1.2.0	Biostrings_2.68.1
## [83]	miniUI_0.1.1.1	knitr_1.43
## [85]	futile.logger_1.4.3	xfun_0.39
## [87]	DT_0.29	stringi_1.7.12
## [89]	yaml_2.3.7	evaluate_0.21
## [91]	codetools_0.2-19	cli_3.6.1
## [93]	survivalROC_1.0.3.1	xtable_1.8-4
## [95]	Rcpp_1.0.10	GenomicDataCommons_1.24.2
## [97]	rmeta_3.0	globals_0.16.2
## [99]	dbplyr_2.3.3	png_0.1-8
## [101]	XML_3.99-0.14	rapiclient_0.1.3
## [103]	parallel_4.3.0	TCGAutils_1.20.2
## [105]	ellipsis_0.3.2	readr_2.1.4
## [107]	blob_1.2.4	prettyunits_1.1.1
## [109]	mclust_6.0.0	bitops_1.0-7
## [111]	listenv_0.9.0	purrr_1.0.1
## [113]	crayon_1.5.2	rlang_1.1.1
## [115]	rvest_1.0.3	KEGGREST_1.40.0
## [117]	formatR_1.14	