Introduction into machine learning and analysis of Breast Cancer Proteomes

Mats Slik

Student: Mats Slik

Student number: 344216

Class: BFV3

Study: Bioinformatics

Institute: Institute for Life Science & Technology

Teachers: Dave Langers (LADR) and Bart Barnard (BABA)

Date: 2022-10-25

Contents

1 Dataset **2**

1.1 About Dataset . . . 2

information about the data set and the three given files : . . . 2

1.2 Exploratory Data Analysis . . . 2

codebook . . . 8

1.3 Data observation . . . 10

1.4 Data cleaning and altering . . . 10

altering sample names . . . 10

numerical data frame . . . 11

Transposing . . . 11

cleaning . . . 11

Merging clinical and protein expression dataframes . . . 14

1.5 Data visualisation . . . 14

2 machine learning **16**

2.1 Weka . . . 18

Data imbalance . . . 18

2.2 Models . . . 18

1 Dataset

1.1 About Dataset

information about the data set and the three given files :

Context: This data set contains published iTRAQ proteome profiling of 77 breast cancer samples generated by the Clinical Proteomic Tumor Analysis Consortium (NCI/NIH). It contains expression values for ~12.000 proteins for each sample, with missing values present when a given protein could not be quantified in a given sample. this data was sampled from 105 originally from the TCGA (The Cancer Genome Atlas Program - NCI), which was further filterd to 77 samples containing high quality protein expression data. **Content:**

- **File:** 77cancerproteomesCPTACitraq.csv
  - **RefSeqaccessionnumber:** RefSeq protein ID (each protein has a unique ID in a RefSeq database)
  - **gene_symbol:** a symbol unique to each gene (every protein is encoded by some gene)
  - **gene_name:** a full name of that gene
  - **Remaining columns:** log2 iTRAQ ratios for each sample (protein expression data, most important), three last columns are from healthy individuals
- **File:** clinicaldatabreast_cancer.csv
  - **First column** "Complete TCGA ID" is used to match the sample IDs in the main cancer proteomes file (see example script).
  - **All other columns** have self-explanatory names, contain data about the cancer classification of a given sample using different methods. 'PAM50 mRNA' classification is being used in the example script.
- **File:** PAM50_proteins.csv
  - **Contains** the list of genes and proteins used by the PAM50 classification system. The column RefSeqProteinID contains the protein IDs that can be matched with the IDs in the main protein expression data set.

Past Research: Original research paper: https://www.researchgate.net/publication/303509927_Proteogenomics_connects_somatic_mutations_to_signaling_in_breast_cancer

Summary: the data were used to assess how the mutations in the DNA are affecting the protein expression landscape in breast cancer. Genes in our DNA are first transcribed into RNA molecules which then are translated into proteins. Changing the information content of DNA has impact on the behavior of the proteome, which is the main functional unit of cells, taking care of cell division, DNA repair, enzymatic reactions and signaling etc.

my question is: Are there different ways to categorize breast cancer based on protein expression data, with machine learning being used to classify them without using the pam50 proteins?

1.2 Exploratory Data Analysis

loading of the dataframes and showing the successful loading and its dimensions. note only the first 5 columns of "77_cancer_proteomes_CPTAC_itraq.csv" are shown since after column 4 they are the same type.

```r
protein_exp_data <- read.csv(file = "data//77_cancer_proteomes_CPTAC_itraq.csv")
clinical_data <- read.csv(file = "data//clinical_data_breast_cancer.csv")
pam50_protein_data <- read.csv(file = "data/PAM50_proteins.csv")

# showing succeseful loading of data

# only showing first 5 columns of proteomes
pander(head(protein_exp_data[1:5], n = 5))
```

Table 1: Table continues below

RefSeq_accession_number	gene_symbol	gene_name	AO.A12D.01TCGA
NP_958782	PLEC	plectin isoform 1	1.096
NP_958785	NA	plectin isoform 1g	1.111
NP_958786	PLEC	plectin isoform 1a	1.111
NP_000436	NA	plectin isoform 1c	1.108
NP_958781	NA	plectin isoform 1e	1.115

C8.A131.01TCGA
2.61
2.65
2.65
2.646
2.646

```r
pander(head(clinical_data, n=5))
```

Table 3: Table continues below

Complete.TCGA.ID	Gender	Age.at.Initial.Pathologic.Diagnosis	ER.Status
TCGA-A2-A0T2	FEMALE	66	Negative
TCGA-A2-A0CM	FEMALE	40	Negative
TCGA-BH-A18V	FEMALE	48	Negative
TCGA-BH-A18Q	FEMALE	56	Negative
TCGA-BH-A0E0	FEMALE	38	Negative

Table 4: Table continues below

PR.Status	HER2.Final.Status	Tumor	Tumor..T1.Coded	Node	Node.Coded
Negative	Negative	T3	T_Other	N3	Positive
Negative	Negative	T2	T_Other	N0	Negative
Negative	Negative	T2	T_Other	N1	Positive
Negative	Negative	T2	T_Other	N1	Positive
Negative	Negative	T3	T_Other	N3	Positive

Table 5: Table continues below

Metastasis	Metastasis.Coded	AJCC.Stage	Converted.Stage
M1	Positive	Stage IV	No_Conversion
M0	Negative	Stage IIA	Stage IIA
M0	Negative	Stage IIB	No_Conversion
M0	Negative	Stage IIB	No_Conversion
M0	Negative	Stage IIIC	No_Conversion

Table 6: Table continues below

Survival.Data.Form	Vital.Status	Days.to.Date.of.Last.Contact
followup	DECEASED	240
followup	DECEASED	754
enrollment	DECEASED	1555
enrollment	DECEASED	1692
followup	LIVING	133

Table 7: Table continues below

Days.to.date.of.Death	OS.event	OS.Time	PAM50.mRNA
240	1	240	Basal-like
754	1	754	Basal-like
1555	1	1555	Basal-like
1692	1	1692	Basal-like
NA	0	133	Basal-like

Table 8: Table continues below

SigClust.Unsupervised.mRNA	SigClust.Intrinsic.mRNA	miRNA.Clusters
0	-13	3
-12	-13	4
-12	-13	5
-12	-13	5
0	-13	5

Table 9: Table continues below

methylation.Clusters	RPPA.Clusters	CN.Clusters
5	Basal	3
4	Basal	4
5	Basal	1
5	Basal	1
5	Basal	1

Table 10: Table continues below

Integrated.Clusters..with.PAM50.	Integrated.Clusters..no.exp.
2	2
2	1
2	2
2	2
2	2

Integrated.Clusters..unsup.exp.
2
1
2
2
2

```
pander(head(pam50_protein_data, n=5))
```

GeneSymbol	RefSeqProteinID	Species	Gene.Name
MIA	NP_006524	Homo sapiens	melanoma inhibitory activity
FGFR4	NP_002002	Homo sapiens	fibroblast growth factor receptor 4
FGFR4	NP_998812	Homo sapiens	fibroblast growth factor receptor 4
FGFR4	NP_075252	Homo sapiens	fibroblast growth factor receptor 4
GPR160	NP_055188	Homo sapiens	G protein-coupled receptor 160

```
# showing the structure/dimensions of dataframe
cat("77_cancer_proteomes_CPTAC_itraq [ number of rows:", nrow(protein_exp_data), "number of columns:", n
```

```
## 77_cancer_proteomes_CPTAC_itraq [ number of rows: 12553 number of columns: 86
```

```
cat("clinical_data [ number of rows:", nrow(clinical_data), "number of columns:",ncol(clinical_data),'\n
```

```
## clinical_data [ number of rows: 105 number of columns: 30
```

```
cat("pam50_protein_data [ number of rows:", nrow(pam50_protein_data), "number of columns:",ncol(pam50_pr
```

```
## pam50_protein_data [ number of rows: 100 number of columns: 4
```

Everything seems to be loaded completely, but we shall look further if everything is also correctly interpreted in R

Now checking if the protein expression data has been correctly read.

```
str(protein_exp_data)
```

```
## 'data.frame':    12553 obs. of  86 variables:
##  $ RefSeq_accession_number: chr  "NP_958782" "NP_958785" "NP_958786" "NP_000436" ...
##  $ gene_symbol            : chr  "PLEC" NA "PLEC" NA ...
##  $ gene_name              : chr  "plectin isoform 1" "plectin isoform 1g" "plectin isoform 1a" "plec
##  $ AO.A12D.01TCGA         : num  1.1 1.11 1.11 1.11 1.12 ...
##  $ C8.A131.01TCGA         : num  2.61 2.65 2.65 2.65 2.65 ...
##  $ AO.A12B.01TCGA         : num  -0.66 -0.649 -0.654 -0.632 -0.64 ...
##  $ BH.A18Q.02TCGA         : num  0.195 0.215 0.215 0.205 0.215 ...
##  $ C8.A130.02TCGA         : num  -0.494 -0.504 -0.501 -0.51 -0.504 ...
##  $ C8.A138.03TCGA         : num  2.77 2.78 2.78 2.8 2.79 ...
##  $ E2.A154.03TCGA         : num  0.863 0.87 0.87 0.866 0.87 ...
##  $ C8.A12L.04TCGA         : num  1.41 1.41 1.41 1.41 1.41 ...
##  $ A2.A0EX.04TCGA         : num  1.19 1.19 1.19 1.19 1.2 ...
##  $ AO.A12D.05TCGA         : num  1.1 1.1 1.1 1.1 1.09 ...
##  $ AN.A04A.05TCGA         : num  0.385 0.371 0.371 0.378 0.375 ...
##  $ BH.A0AV.05TCGA         : num  0.351 0.367 0.367 0.361 0.371 ...
##  $ C8.A12T.06TCGA         : num  -0.205 -0.162 -0.167 -0.184 -0.167 ...
##  $ A8.A06Z.07TCGA         : num  -0.496 -0.499 -0.496 -0.492 -0.488 ...
##  $ A2.A0CM.07TCGA         : num  0.683 0.694 0.698 0.687 0.687 ...
##  $ BH.A18U.08TCGA         : num  -0.265 -0.252 -0.252 -0.252 -0.252 ...
##  $ A2.A0EQ.08TCGA         : num  -0.913 -0.928 -0.928 -0.932 -0.928 ...
##  $ AR.A0U4.09TCGA         : num  -0.0332 -0.0302 -0.0272 -0.0302 -0.0302 ...
##  $ AO.A0J9.10TCGA         : num  0.02 0.012 0.012 0.0039 0.012 ...
##  $ AR.A1AP.11TCGA         : num  0.461 0.461 0.461 0.461 0.461 ...
##  $ AN.A0FK.11TCGA         : num  0.974 0.977 0.977 0.97 0.985 ...
##  $ AO.A0J6.11TCGA         : num  0.831 0.857 0.857 0.837 0.865 ...
##  $ A7.A13F.12TCGA         : num  1.28 1.28 1.28 1.28 1.28 ...
##  $ BH.A0E1.12TCGA         : num  0.762 0.762 0.766 0.758 0.766 ...
##  $ A7.A0CE.13TCGA         : num  -1.12 -1.12 -1.12 -1.13 -1.13 ...
##  $ A2.A0YC.13TCGA         : num  0.819 0.815 0.815 0.799 0.819 ...
##  $ AO.A0JC.14TCGA         : num  -0.307 -0.307 -0.307 -0.307 -0.301 ...
##  $ A8.A08Z.14TCGA         : num  0.569 0.569 0.569 0.569 0.569 ...
##  $ AR.A0TX.14TCGA         : num  -0.583 -0.573 -0.567 -0.583 -0.573 ...
##  $ A8.A076.15TCGA         : num  1.87 1.87 1.87 1.86 1.87 ...
##  $ AO.A126.15TCGA         : num  0.196 0.196 0.196 0.219 0.2 ...
##  $ BH.A0C1.16TCGA         : num  -0.518 -0.51 -0.507 -0.518 -0.513 ...
##  $ A2.A0EY.16TCGA         : num  1.17 1.18 1.18 1.17 1.18 ...
##  $ AR.A1AW.17TCGA         : num  0.578 0.582 0.578 0.59 0.586 ...
##  $ AR.A1AV.17TCGA         : num  -0.76 -0.76 -0.749 -0.736 -0.749 ...
##  $ C8.A135.17TCGA         : num  1.12 1.14 1.14 1.14 1.12 ...
##  $ A2.A0EV.18TCGA         : num  0.453 0.473 0.473 0.459 0.473 ...
##  $ AN.A0AM.18TCGA         : num  1.5 1.51 1.5 1.5 1.5 ...
##  $ D8.A142.18TCGA         : num  0.539 0.542 0.542 0.535 0.542 ...
##  $ AN.A0FL.19TCGA         : num  2.46 2.48 2.48 2.46 2.48 ...
##  $ BH.A0DG.19TCGA         : num  -0.206 -0.206 -0.206 -0.215 -0.206 ...
##  $ AR.A0TV.20TCGA         : num  -1.51 -1.53 -1.53 -1.53 -1.51 ...
##  $ C8.A12Z.20TCGA         : num  -0.787 -0.756 -0.756 -0.775 -0.772 ...
##  $ AO.A0JJ.20TCGA         : num  0.757 0.781 0.774 0.764 0.771 ...
##  $ AO.A0JE.21TCGA         : num  0.56 0.563 0.56 0.542 0.56 ...
##  $ AN.A0AJ.21TCGA         : num  -0.428 -0.406 -0.406 -0.406 -0.406 ...
##  $ A7.A0CJ.22TCGA         : num  -1.001 -1.005 -1.005 -0.998 -1.001 ...
##  $ AO.A12F.22TCGA         : num  -1.95 -1.95 -1.96 -1.95 -1.96 ...
##  $ A8.A079.23TCGA         : num  1.05 1.05 1.05 1.06 1.05 ...
##  $ A2.A0T3.24TCGA         : num  0.584 0.581 0.581 0.587 0.587 ...
```

```
##  $ A2.A0YD.24TCGA         : num  0.0638 0.0933 0.0845 0.0667 0.0845 ...
##  $ AR.A0TR.25TCGA         : num  -1.1 -1.11 -1.11 -1.1 -1.11 ...
##  $ AO.A03O.25TCGA         : num  1.05 1.06 1.06 1.06 1.06 ...
##  $ AO.A12E.26TCGA         : num  0.265 0.276 0.276 0.278 0.278 ...
##  $ A8.A06N.26TCGA         : num  0.239 0.25 0.244 0.25 0.25 ...
##  $ A2.A0YG.27TCGA         : num  -0.0782 -0.0681 -0.0714 -0.0579 -0.0647 ...
##  $ BH.A18N.27TCGA         : num  1.1 1.1 1.1 1.09 1.11 ...
##  $ AN.A0AL.28TCGA         : num  0.324 0.327 0.327 0.33 0.327 ...
##  $ A2.A0T6.29TCGA         : num  0.794 0.818 0.815 0.801 0.818 ...
##  $ E2.A158.29TCGA         : num  -1.09 -1.1 -1.1 -1.1 -1.1 ...
##  $ E2.A15A.29TCGA         : num  2.18 2.18 2.18 2.18 2.18 ...
##  $ AO.A0JM.30TCGA         : num  1.4 1.41 1.41 1.41 1.41 ...
##  $ C8.A12V.30TCGA         : num  0.674 0.689 0.689 0.678 0.689 ...
##  $ A2.A0D2.31TCGA         : num  0.1075 0.1042 0.1075 0.0975 0.1042 ...
##  $ C8.A12U.31TCGA         : num  -0.482 -0.478 -0.482 -0.471 -0.482 ...
##  $ AR.A1AS.31TCGA         : num  1.22 1.22 1.22 1.2 1.22 ...
##  $ A8.A09G.32TCGA         : num  -1.52 -1.51 -1.51 -1.52 -1.51 ...
##  $ C8.A131.32TCGA         : num  2.71 2.73 2.74 2.73 2.75 ...
##  $ C8.A134.32TCGA         : num  0.14 0.126 0.133 0.112 0.126 ...
##  $ A2.A0YF.33TCGA         : num  0.311 0.296 0.296 0.296 0.296 ...
##  $ BH.A0DD.33TCGA         : num  -0.692 -0.659 -0.664 -0.657 -0.662 ...
##  $ BH.A0E9.33TCGA         : num  1.47 1.48 1.47 1.46 1.47 ...
##  $ AR.A0TT.34TCGA         : num  -0.511 -0.526 -0.526 -0.533 -0.53 ...
##  $ AO.A12B.34TCGA         : num  -0.964 -0.938 -0.944 -0.935 -0.935 ...
##  $ A2.A0SW.35TCGA         : num  -0.488 -0.488 -0.488 -0.488 -0.504 ...
##  $ AO.A0JL.35TCGA         : num  -0.107 -0.107 -0.107 -0.107 -0.107 ...
##  $ BH.A0BV.35TCGA         : num  -0.0658 -0.0559 -0.0658 -0.0559 -0.0625 ...
##  $ A2.A0YM.36TCGA         : num  0.656 0.658 0.656 0.656 0.651 ...
##  $ BH.A0C7.36TCGA         : num  -0.552 -0.548 -0.552 -0.552 -0.557 ...
##  $ A2.A0SX.36TCGA         : num  -0.399 -0.393 -0.393 -0.393 -0.396 ...
##  $ X263d3f.I.CPTAC        : num  0.599 0.607 0.604 0.604 0.604 ...
##  $ blcdb9.I.CPTAC         : num  -0.191 -0.184 -0.186 -0.186 -0.167 ...
##  $ c4155b.C.CPTAC         : num  0.567 0.579 0.577 0.577 0.577 ...
```

Nothing strange about the Proteomes dat everything seems to be read correct.

Checking if the clinical data has been correctly read.

```
str(clinical_data)
```

```
## 'data.frame':    105 obs. of  30 variables:
##  $ Complete.TCGA.ID                   : chr  "TCGA-A2-A0T2" "TCGA-A2-A0CM" "TCGA-BH-A18V" "TCGA-BH-A
##  $ Gender                             : chr  "FEMALE" "FEMALE" "FEMALE" "FEMALE" ...
##  $ Age.at.Initial.Pathologic.Diagnosis: int  66 40 48 56 38 57 74 60 61 67 ...
##  $ ER.Status                          : chr  "Negative" "Negative" "Negative" "Negative" ...
##  $ PR.Status                          : chr  "Negative" "Negative" "Negative" "Negative" ...
##  $ HER2.Final.Status                  : chr  "Negative" "Negative" "Negative" "Negative" ...
##  $ Tumor                              : chr  "T3" "T2" "T2" "T2" ...
##  $ Tumor..T1.Coded                    : chr  "T_Other" "T_Other" "T_Other" "T_Other" ...
##  $ Node                               : chr  "N3" "N0" "N1" "N1" ...
##  $ Node.Coded                         : chr  "Positive" "Negative" "Positive" "Positive" ...
##  $ Metastasis                         : chr  "M1" "M0" "M0" "M0" ...
##  $ Metastasis.Coded                   : chr  "Positive" "Negative" "Negative" "Negative" ...
##  $ AJCC.Stage                         : chr  "Stage IV" "Stage IIA" "Stage IIB" "Stage IIB" ...
```

```
##  $ Converted.Stage                    : chr  "No_Conversion" "Stage IIA" "No_Conversion" "No_Conversi
##  $ Survival.Data.Form                 : chr  "followup" "followup" "enrollment" "enrollment" ...
##  $ Vital.Status                       : chr  "DECEASED" "DECEASED" "DECEASED" "DECEASED" ...
##  $ Days.to.Date.of.Last.Contact       : int  240 754 1555 1692 133 309 425 643 775 964 ...
##  $ Days.to.date.of.Death              : int  240 754 1555 1692 NA NA NA NA NA NA ...
##  $ OS.event                           : int  1 1 1 1 0 0 0 0 0 0 ...
##  $ OS.Time                            : int  240 754 1555 1692 133 309 425 643 775 964 ...
##  $ PAM50.mRNA                         : chr  "Basal-like" "Basal-like" "Basal-like" "Basal-like" ...
##  $ SigClust.Unsupervised.mRNA         : int  0 -12 -12 -12 0 0 0 -12 -12 -12 ...
##  $ SigClust.Intrinsic.mRNA            : int  -13 -13 -13 -13 -13 -13 -13 -13 -13 -13 ...
##  $ miRNA.Clusters                     : int  3 4 5 5 5 5 3 5 2 5 ...
##  $ methylation.Clusters               : int  5 4 5 5 5 5 5 5 5 5 ...
##  $ RPPA.Clusters                      : chr  "Basal" "Basal" "Basal" "Basal" ...
##  $ CN.Clusters                        : int  3 4 1 1 1 1 1 1 1 3 ...
##  $ Integrated.Clusters..with.PAM50.   : int  2 2 2 2 2 2 2 2 2 2 ...
##  $ Integrated.Clusters..no.exp.       : int  2 1 2 2 2 2 2 2 2 2 ...
##  $ Integrated.Clusters..unsup.exp.    : int  2 1 2 2 2 2 2 2 2 2 ...
```

Nothing strange about the clinical data everything seems to be read correct.

Checking if the pam50 protein data has been correctly read.

```
str(pam50_protein_data)
```

```
## 'data.frame':    100 obs. of  4 variables:
##  $ GeneSymbol     : chr  "MIA" "FGFR4" "FGFR4" "FGFR4" ...
##  $ RefSeqProteinID: chr  "NP_006524" "NP_002002" "NP_998812" "NP_075252" ...
##  $ Species        : chr  "Homo sapiens" "Homo sapiens" "Homo sapiens" "Homo sapiens" ...
##  $ Gene.Name      : chr  "melanoma inhibitory activity" "fibroblast growth factor receptor 4" "fibrol
```

Nothing strange about the pam50 protein data everything seems to be read correct.

codebook

loading of the created codebooks for the three dataframes. showing also its contents and successful loading

```
cancer_proteomes_CPTAC_codebook <- read.csv2("data/77_cancer_proteomes_CPTAC_codebook.txt")
clinical_data_codebook <- read.csv2("data/clinical_data_breast_cancer_codebook.txt")
PAM50_protein_codebook <- read.csv2("data/PAM50_protein_codebook.txt", sep = ";")


pander(cancer_proteomes_CPTAC_codebook)
```

Column	Description	data.type	unit
RefSeq_accession_number	RefSeq protein ID	string	NA
gene_symbol	Gene abbreviation code	string	NA
gene_name	Name of the gene	string	NA
Remaining columns	log2 iTRAQ ratios	float	NA

```
pander(clinical_data_codebook)
```

Table 14: Table continues below

Column	Description
Complete_TCGA_ID	TCGA ID
Gender	Gender
Age_at_Initial_Pathologic_Diagnosis	Age at Initial Pathologic Diagnosis
ER Status	Estrogen receptor Status
PR Status	Progesterone receptor Status
HER2 Final Status	Human Epidermal growth factor Receptor 2
Tumor	Tumor
Tumor–T1 Coded	Tumor–T1 Coded
Node	Node
Node-Coded	Node-Coded
Metastasis	Metastasis
Metastasis-Coded	Metastasis-Coded
AJCC Stage	American Joint Committee on Cancer Stage
Converted Stage	Converted Stage
Survival Data Form	Survival Data Form
Vital Status	Vital Status
Days to Date of Last Contact	Days to Date of Last Contact
Days to date of Death	Days to date of Death
OS event	OS event 0= NO, 1= YES
OS Time	OS Time
PAM50 mRNA	PAM50 mRNA
SigClust Unsupervised mRNA	SigClust Unsupervised mRNA
SigClust Intrinsic mRNA	SigClust Intrinsic mRNA
miRNA Clusters	miRNA Clusters
methylation Clusters	methylation Clusters
RPPA Clusters	RPPA Clusters
CN Clusters	CN Clusters
Integrated Clusters (with PAM50)	Integrated Clusters (with PAM50)
Integrated Clusters (no exp)	Integrated Clusters (no exp)
Integrated Clusters (unsup exp)	Integrated Clusters (unsup exp)

type	data.type	unit
name	chr	NA
name	chr	NA
Descriptive	int	Years
Descriptive	chr	NA
Descriptive	chr	NA
Descriptive	chr	NA
Descriptive	chr	NA
Descriptive	chr	NA
Descriptive	chr	NA
Descriptive	chr	NA
Descriptive	chr	NA
Descriptive	chr	NA
Descriptive	chr	NA
Descriptive	chr	NA

type	data.type	unit
Descriptive	chr	NA
Descriptive	chr	NA
Descriptive	chr	NA
Time	int	Days
Time	int	Days
Descriptive	int	NA
Time	int	Hours
Descriptive	chr	NA
Count	int	NA
Count	int	NA
Count	int	NA
Count	int	NA
Descriptive	chr	NA
Count	int	NA
Count	int	NA
Count	int	NA
Count	int	NA

```
pander(PAM50_protein_codebook)
```

Column	Description	type	unit
GeneSymbol	Gene abbreviation	chr	NA
RefSeqProteinID	Unique reference identifier	chr	NA
Species	Species	chr	latin name
Gene.Name	Name of the gene	chr	NA

Here we can also see that everything has been successfully loaded into R

1.3 Data observation

there are 12553 rows in the data, these are proteins identifiable with a RefSeq ID number and have 86 columns of which the last 83 are samples(with their identifiers as there name and the last three from healthy individuals, but these shall not be used for the machine learning part since 3 samples is too little to use for analyzes. to further use the data I shall reshape it to make the rows samples and each column a protein.

1.4 Data cleaning and altering

altering sample names

the alteration of sample names to corespondent to the clinical data names is needed for further comparison and analyses. this is done by changing the column names to that of the same format of the clinical data. this is done with some regex magic.

```
# storing a list of the column names
column_names <- names(protein_exp_data)

# function
```

```
change_sample_name <- function (x){
  #search for TCGA name,if found split and make new name
    if(grepl("TCGA",x) == TRUE){
      temp_list <- as.list(strsplit(x, '[_|-|.]')[[1]])
      x <- str_c(c('TCGA',temp_list[[1]],temp_list[[2]]),collapse = '-')
    }
return (x)
}

# changing of the colnames
colnames(protein_exp_data) <- lapply(column_names, change_sample_name)
cat("Old name:",column_names[[4]],",New name:",names(protein_exp_data)[[4]])
```

```
## Old name: AO.A12D.01TCGA ,New name: TCGA-AO-A12D
```

this output show to conversion has been successful

numerical data frame

now we need to make a data frame with only the numerical data for the ease of analyzes

```
# first making a data frame with only the numerical data, samples start at column number 4 til the end
protein_exp_numerical <- protein_exp_data[4:86]
```

Transposing

transposing the created data frame "protein_exp_numerical", and adding the refseq ID as column name

```
# transposing of the old dataframe to a new one
protein_exp__numerical_transposed <- as.data.frame(t(protein_exp_numerical))
colnames(protein_exp__numerical_transposed) <- protein_exp_data$RefSeq_accession_number

# checking if succesfull
cat("protein_exp_numerical number of rows:", nrow(protein_exp_numerical),
    "number of columns:", ncol(protein_exp_numerical), '\n')
```

```
## protein_exp_numerical number of rows: 12553 number of columns: 83
```

```
cat("protein_exp__numerical_transposed number of rows:", nrow(protein_exp__numerical_transposed),
    "number of columns:", ncol(protein_exp__numerical_transposed), '\n')
```

```
## protein_exp__numerical_transposed number of rows: 83 number of columns: 12553
```

as we can see the row and column dimensions have been flipped

cleaning

since there are NA values in the data lets see how much

```
count_na_func <- function(x) sum(is.na(x))
# getting NA values per RefSeqID(column)
Na_per_col <- sapply(protein_exp__numerical_transposed, count_na_func)
```

```
ggplot() +
  aes(Na_per_col) +
  geom_histogram(color = "black", fill = "#F9C000", binwidth = 3) +
  xlab("Number of NA") +
  ylab("Frequency of columns") +
  ggtitle("Frequency of number of NA values per RefSeqID")
```

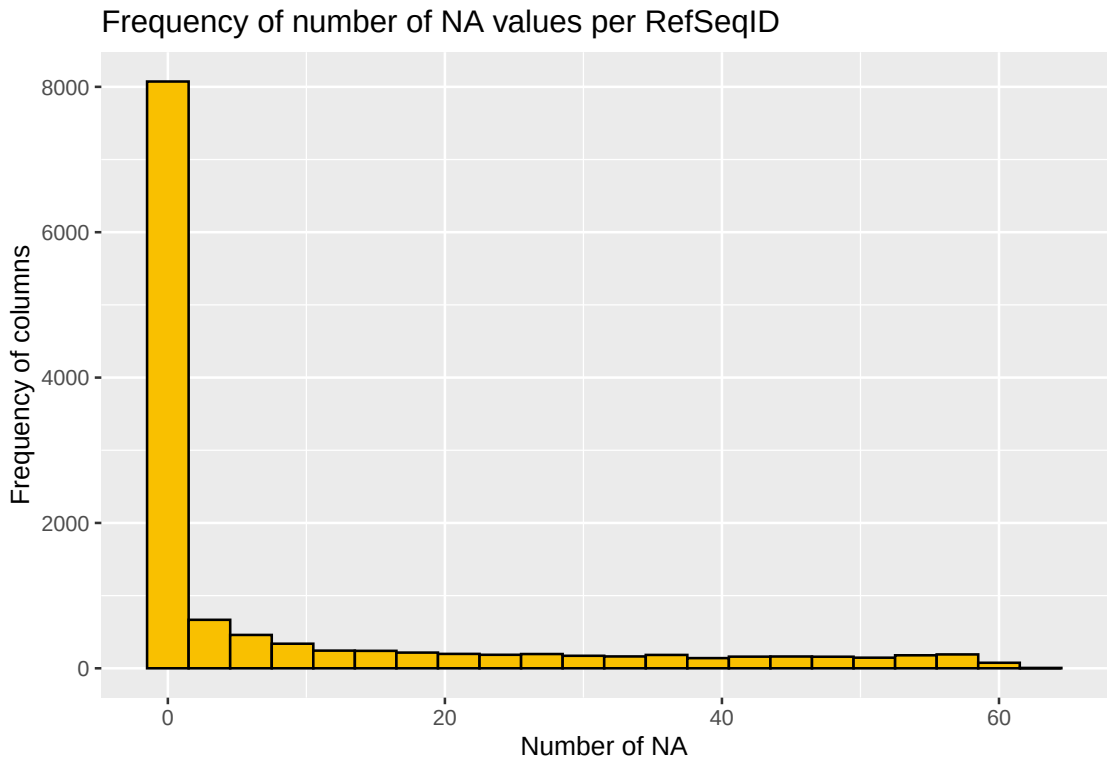

Figure 1: barplot with the frequency of columns with more than 0 NA in them

```
ggsave(
  filename = "figure1.png",
  plot = last_plot(),
  path = "../data/figures")
```

```
## Saving 6.5 x 4.5 in image
```

```
ggplot() +
  aes(Na_per_col[Na_per_col > 0]) +
  geom_histogram(color = "black", fill = "#0039F9",binwidth = 3) +
  xlab("Number of NA") +
  ylab("Frequency of columns") +
  ggtitle("Frequency of number of NA values per RefSeqID with 0 omitted")
```

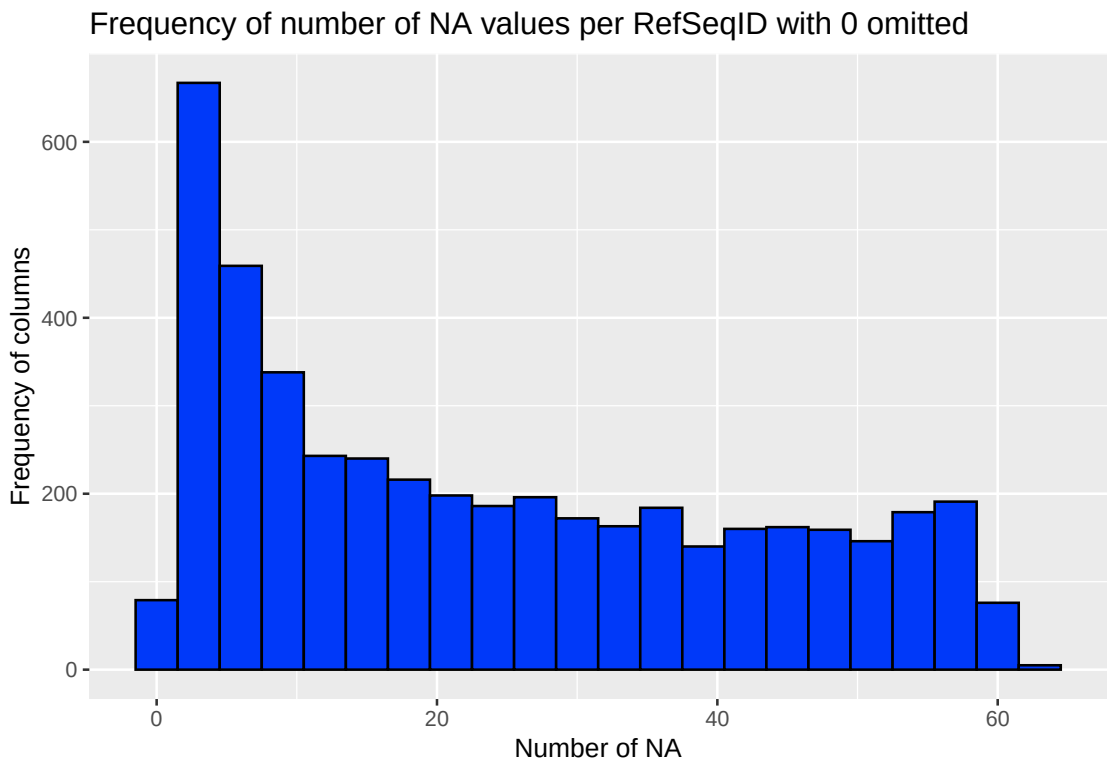

Figure 2: barplot with the frequency of columns with more than 0 NA in them

```
ggsave(
  filename = "figure2.png",
  plot = last_plot(),
  path = "../data/figures")
```

```
## Saving 6.5 x 4.5 in image
```

there are a lot of proteins with more than 10% of their samples with missing data, so i shall be removing.

```
cat("number of proteins with NA values in them:", sum(Na_per_col > 0), '\n' )
```

```
## number of proteins with NA values in them: 4559
```

```
# deleting every protein with more than 10% NA values in them, since this is allot
proteomes_filtered_data <- protein_exp__numerical_transposed[Na_per_col < 8]
cat("number of proteins with 8 or more NA values in them and deleted from data:",
    sum(Na_per_col > 8), '\n')
```

```
## number of proteins with 8 or more NA values in them and deleted from data: 3219
```

```
cat("proteomes_filtered_data[number of rows:",
    nrow(proteomes_filtered_data),
    "number of columns:",
    ncol(proteomes_filtered_data),'\n')
```

```
## proteomes_filtered_data[number of rows: 83 number of columns: 9199
```

as we can see from te reports generated by the code we can see that the filtering of NA was successful. and we now have a data set that contains data with less than 10% per protein of NA values

Merging clinical and protein expression dataframes

to be able to use the Clinical dat we need to merge it to its corresponding row and sample in the Protein expressions

```
# firs assigning row names to clinical data
rownames(clinical_data) <- clinical_data$Complete.TCGA.ID


# removing the used ID column to simplifie it since it has been become redundend
clinical_data <- clinical_data[,-1]

# mergin the two data frames according to the row names (TCGA Identification number),
merged_data <- merge(select(clinical_data, Tumor, Tumor..T1.Coded, AJCC.Stage, Vital.Status), protein_e
cleaned_merged_data <- merge(select(clinical_data, Tumor, Tumor..T1.Coded, AJCC.Stage, Vital.Status), p
cat("merged_data of rows:", nrow(merged_data),
    "number of columns:",ncol(merged_data),'\n')
```

```
## merged_data of rows: 77 number of columns: 12558
```

we can see that not every sample had an entry in the clinical data, so we end up with 6 rows of sample data that get left out of the merged data set

1.5 Data visualisation

Showing some examples of distributions of protein expression data since there are around 12 000 proteins found

```
# Open a png file
#png("../data/figures/figure3.png")
# 2. Create a plot
boxplot(merged_data[6:76],
        col = rainbow(ncol(merged_data[6:76])),
                          xlab = "Protein",
                          ylab = "log2 iTRAQ ratio",
                          main = "distribution of Protein expression for first 70 Proteins",
                          show.names= FALSE)
```

```
# Close the pdf file
#dev.off()
```

Since i can't be sure of the significant of every protein I can't simply trow away any sample

to fix this i shall get the standard deviation from every protein to prepare for a selection of the most deviant ones. and for illustration i will overlay the dataframe which has been filtered from most NA values

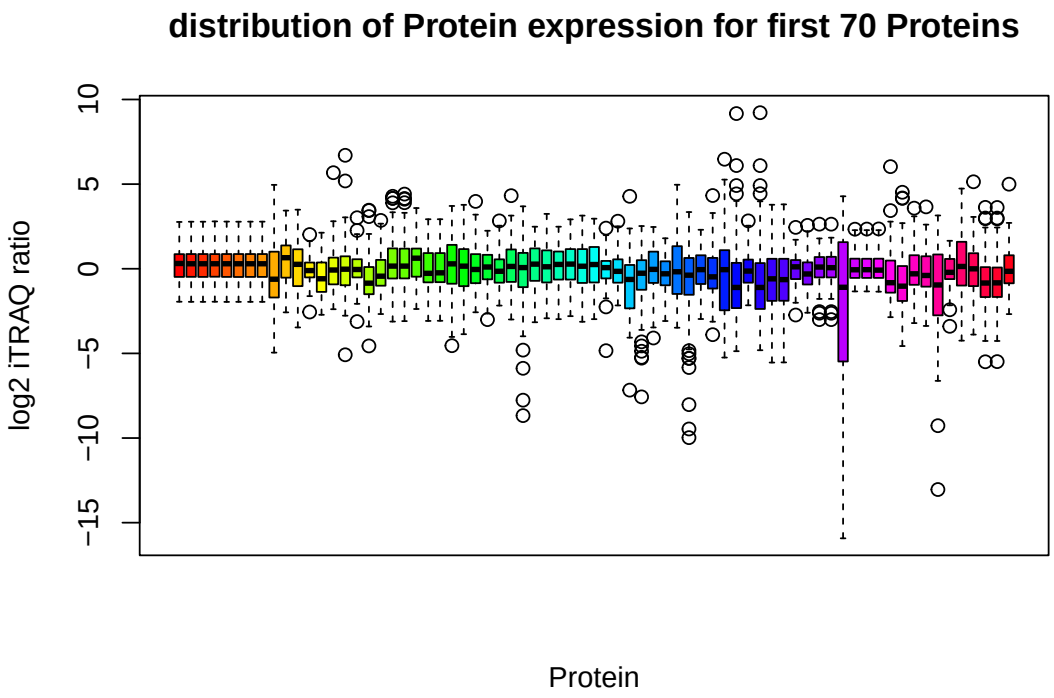

Figure 3: boxplot of protein expression distribution, first 70 proteins

```
fg <- colSds(as.matrix(merged_data[sapply(merged_data, is.numeric)]), na.rm = TRUE)
fg2 <- colSds(as.matrix(cleaned_merged_data[sapply(cleaned_merged_data, is.numeric)]), na.rm = TRUE)


fg <- as.data.frame(fg, row.names = colnames(merged_data[6:ncol(merged_data)]))
fg2 <- as.data.frame(fg2, row.names = colnames(cleaned_merged_data[6:ncol(cleaned_merged_data)]))

fg$count <- seq(from = 1, to = nrow(fg))

merged_fg <- merge(fg,fg2,all.x = TRUE, by = 0)
merged_fg <- merged_fg[order(merged_fg$count),]

# assigning 0 to every NA values in fg2 since that is the filterd one
merged_fg$fg2[is.na(merged_fg$fg2)] <- 0

# plotting
ggplot(data=merged_fg) +
  geom_point(size=0.0015, aes(x=count,  y=fg, color="Non Na filtered")) +
  geom_point(size=0.0015, aes(x=count,y=fg2,  color="Na filtered")) +
  labs(x = "Data Frame row number",
         y = "Standard deviation",
         color = "Legend") +
  ggtitle("Density plot for standard deviation of protein expression") +
  theme(legend.position = "bottom")
```

Figure 4: scatterplot of standard deviation for the protein expresion data and for the NA filterd version

```
ggsave(
  filename = "figure4.png",
  plot = last_plot(),
  path = "../data/figures")
```

```
## Saving 6.5 x 4.5 in image
```

as we can see in the standard deviation compared from proteins with a lot of samples with NA in them to the filtered data, we see that the proteins with a lot of NA in them seem to be having a higher Standard deviation, but still there should be enough deviation in the filtered Data.

```
# assigning as factors
merged_data$Tumor <- factor(merged_data$Tumor)
cleaned_merged_data$Tumor <- factor(cleaned_merged_data$Tumor)
#merged_data$Row.names <- factor(merged_data$Row.names, levels =)
#cleaned_merged_data$Row.names <- factor(cleaned_merged_data$Row.names)
```

```
plot(cleaned_merged_data$Tumor, ylab = "Count", xlab="tumor stage", col ="#F9C000", main = "distributio
```

2 machine learning

In this chapter, we are looking at how we are going to train the machine learning algorithms to accurately predict the tumor stage of breast cancer according to the protein expressions found in breast tissue samples.

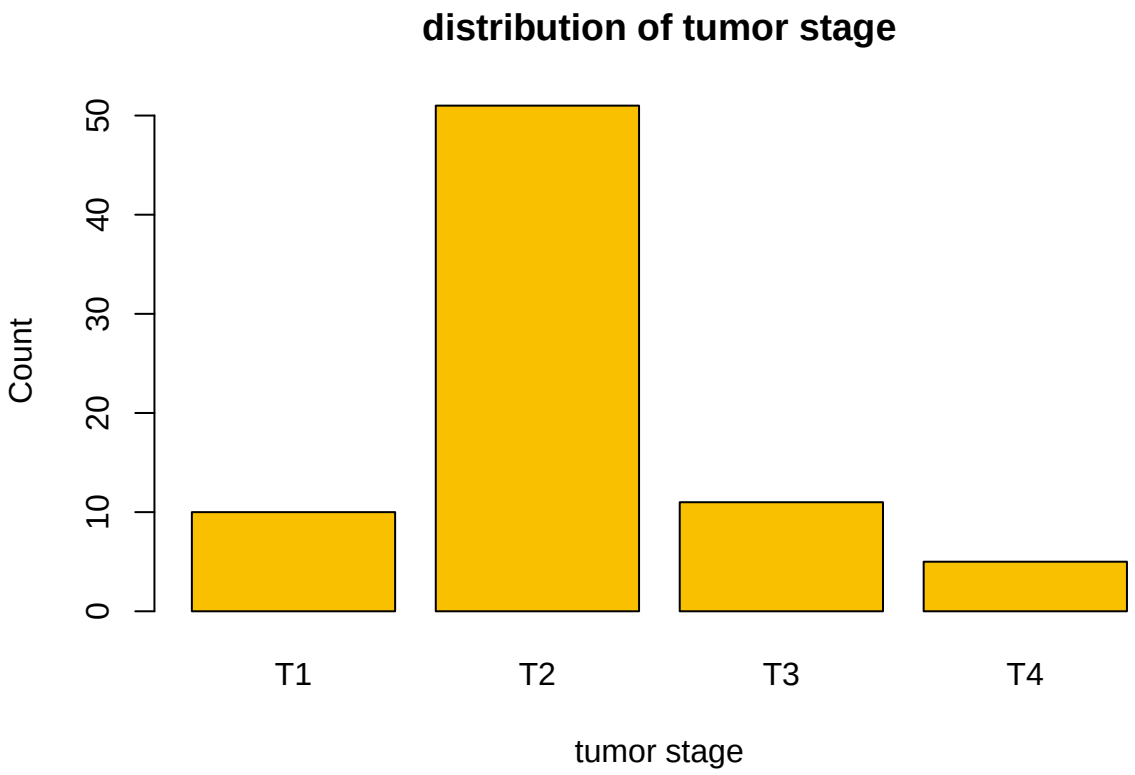

Figure 5: distribution of amount of samples per tumor stage

after that there shall be a examination of the result and accuracy of the created models accordingly to different algorithms.

2.1 Weka

Weka is used for the data examination and machine learning part, Weka is a open source collection of machine learning algorithms for data mining tasks. It contains tools for data preparation, classification, regression, clustering, association rules mining, and visualization. java platform for Firstly the data is exported to a .arff file so it can be loaded into Weka

Data imbalance

```
set.seed(420)


train <- createDataPartition(cleaned_merged_data$Tumor,
                             p = 0.7, # % of data going to training
                             times = 1,
                             list = F)
train.orig <- cleaned_merged_data[ train,]
test       <- cleaned_merged_data[-train,]

train.orig <- train.orig[,c(-1,-3:-5)]
test       <- test[,c(-1,-3:-5)]

train.orig$data.class <- as.factor(train.orig$Tumor)
test$data.class <- as.factor(test$Tumor)
```

```
write.arff(train.orig, file = "data/train.orig.arff")
write.arff(test, file = "data/test_data.arff")
```

2.2 Models