Data Exploration & Cleaning for Machine Learning

Lauren Vanderlinden, PhD, MS
T15 Postdoctoral Fellow Computational Biology
Division of Rheumatology & Department of Biomedical Informatics
School of Medicine, University of Colorado Anschutz Medical Campus

CPBS 7602 - December 12, 2024

Outline

- Basics of data exploration
- Histograms and boxplots
- Outliers
- Missing data
- Prepping data for ML

Exploration Basics – Where does it fit in the process?

Why do we do data exploration?

- Understanding data
- Data preparation

Summary Statistics

Univariate Summary

- Mean, median, standard deviation (and so on) for continuous measures
- N's and percentages for categorical variables
- R/table1 is a great tool I use for this

```
table1(~ Gender + Race + BMI + FirstDegreeRelWithRA + smoke +
visit + IAEver + lipid + lipid2 | as.factor(ra), dat=dat,
overall=TRUE)
```

Table 1

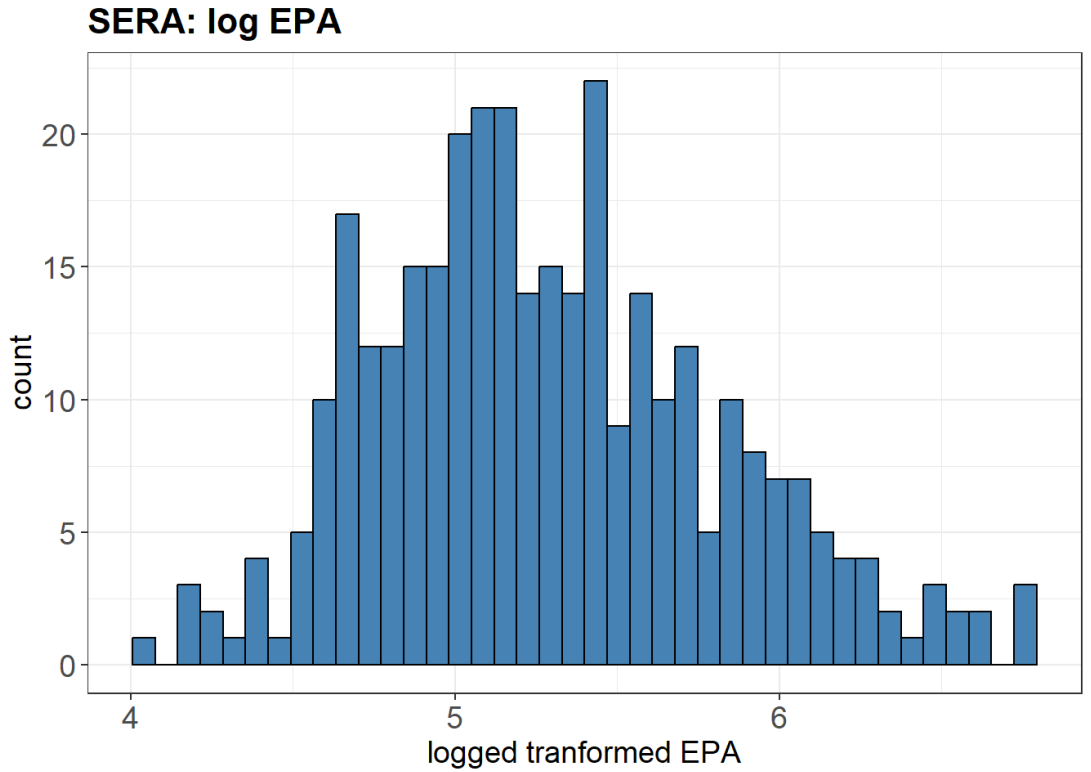
```
table1(~ Gender + Race + BMI + FirstDegreeRelWithRA + smoke +
visit + IAEver + lipid + lipid2 | as.factor(ra), dat=dat,
overall=TRUE)
```

	0 (N=67)	1 (N=20)	Overall (N=87)
Gender			
Female	47 (70.1%)	14 (70.0%)	61 (70.1%)
Male	20 (29.9%)	6 (30.0%)	26 (29.9%)
Race			
Asian	1 (1.5%)	0 (0%)	1 (1.1%)
Biracial	3 (4.5%)	4 (20.0%)	7 (8.0%)
Black	1 (1.5%)	0 (0%)	1 (1.1%)
Hispanic	3 (4.5%)	0 (0%)	3 (3.4%)
NHW	59 (88.1%)	16 (80.0%)	75 (86.2%)
BMI			
Mean (SD)	31.6 (32.8)	27.2 (4.90)	30.6 (28.9)
Median [Min, Max]	26.6 [17.0, 285]	26.2 [20.7, 41.2]	26.5 [17.0, 285]
Missing	3 (4.5%)	1 (5.0%)	4 (4.6%)
FirstDegreeRelWithRA			
No	18 (26.9%)	10 (50.0%)	28 (32.2%)
Yes	49 (73.1%)	10 (50.0%)	59 (67.8%)
smoke			
Mean (SD)	0.471 (0.507)	0.0909 (0.302)	0.378 (0.490)
Median [Min, Max]	0 [0, 1.00]	0 [0, 1.00]	0 [0, 1.00]
Missing	33 (49.3%)	9 (45.0%)	42 (48.3%)
visit			
Mean (SD)	1.03 (0.171)	1.10 (0.308)	1.05 (0.211)
Median [Min, Max]	1.00 [1.00, 2.00]	1.00 [1.00, 2.00]	1.00 [1.00, 2.00]
IAEver			
	63 (94.0%)	0 (0%)	63 (72.4%)
No	4 (6.0%)	0 (0%)	4 (4.6%)
Yes	0 (0%)	20 (100%)	20 (23.0%)
lipid			
Mean (SD)	36200 (32100)	41000 (28400)	37300 (31200)
Median [Min, Max]	26500 [1000, 181000]	33700 [15700, 133000]	27100 [1000, 181000]
lipid2			
Mean (SD)	11000 (13700)	11700 (12400)	11100 (13300)
Median [Min, Max]	5830 [1030, 80300]	9450 [3180, 61100]	6140 [1030, 80300]

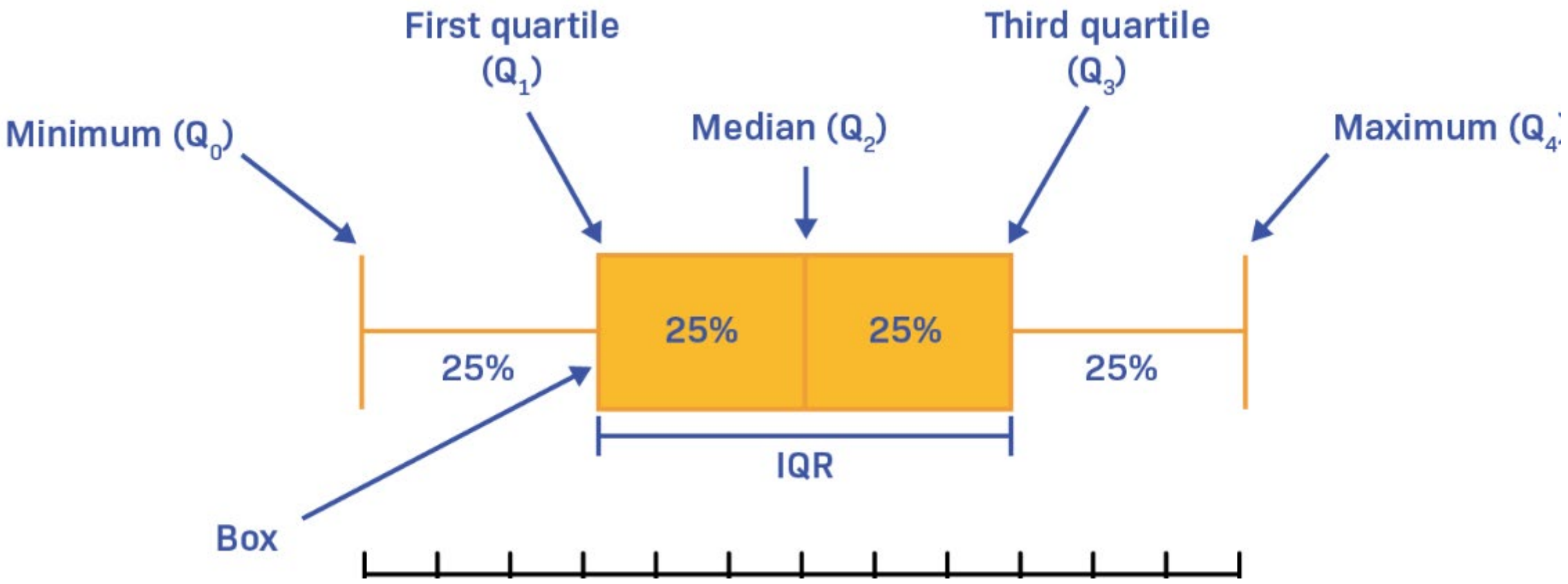
Visuals

Histograms

- It is a representation of a frequency distribution by means of rectangles whose widths represent class intervals and whose areas are proportional to the corresponding frequencies.



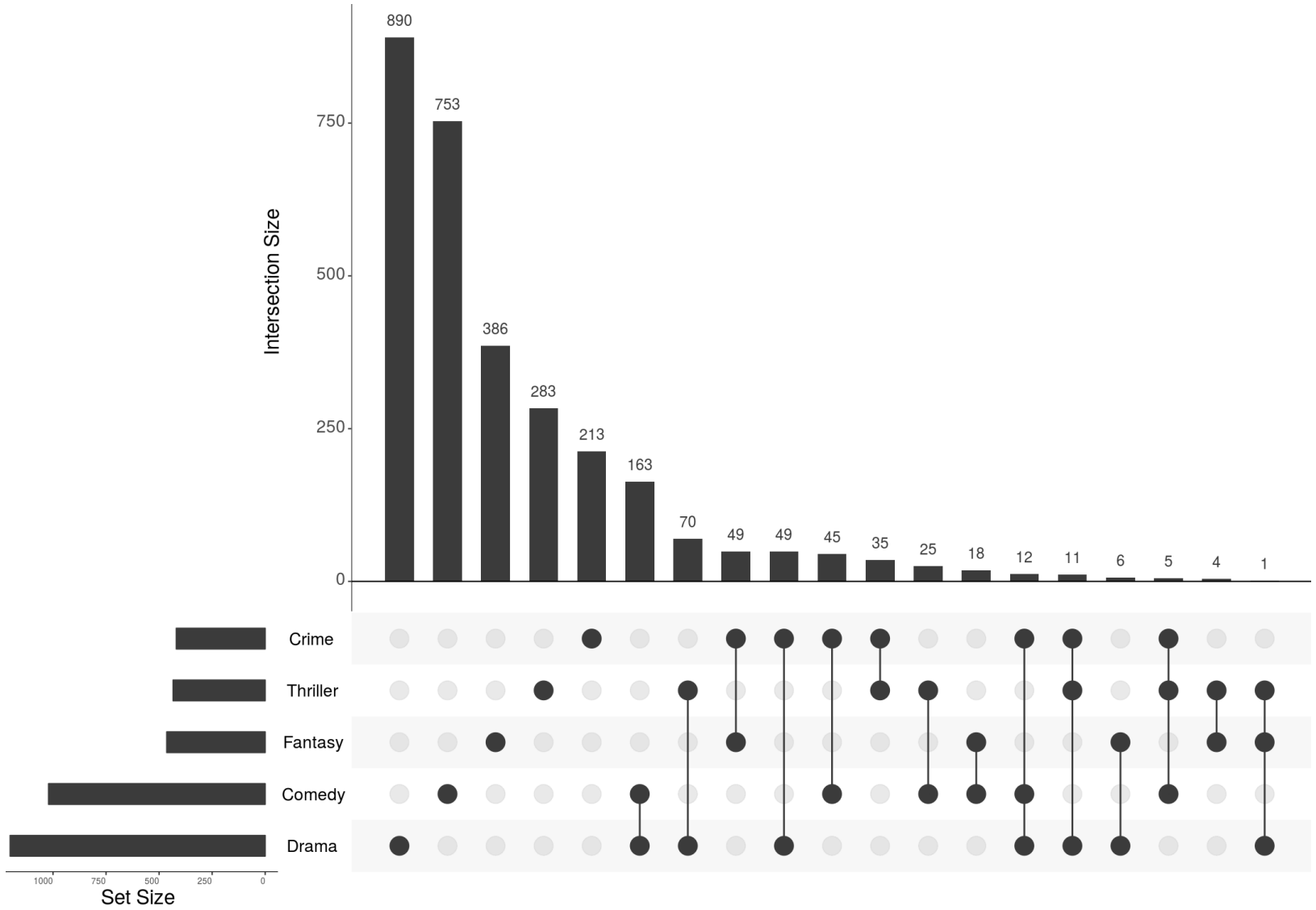
Box plots


First quartile
(Q_1)
Third quartile
(Q_3)
Minimum (Q_0)
Median (Q_2)
Maximum (Q_4)
25%
25%
25%
25%
IQR
Box

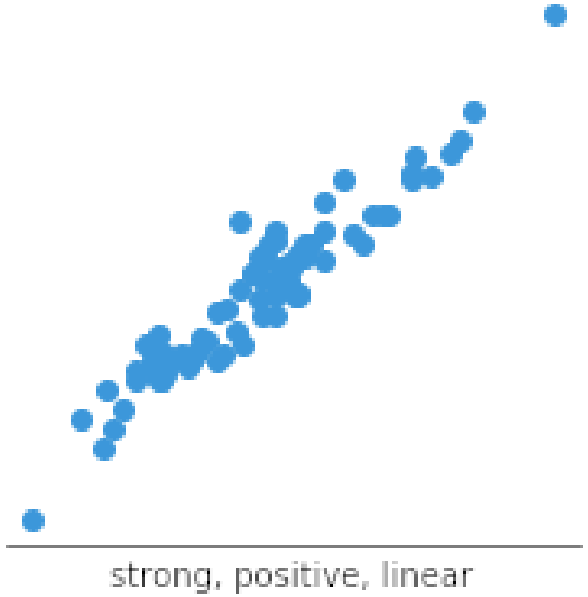
UpSet Plots

Very helpful when looking at data availability

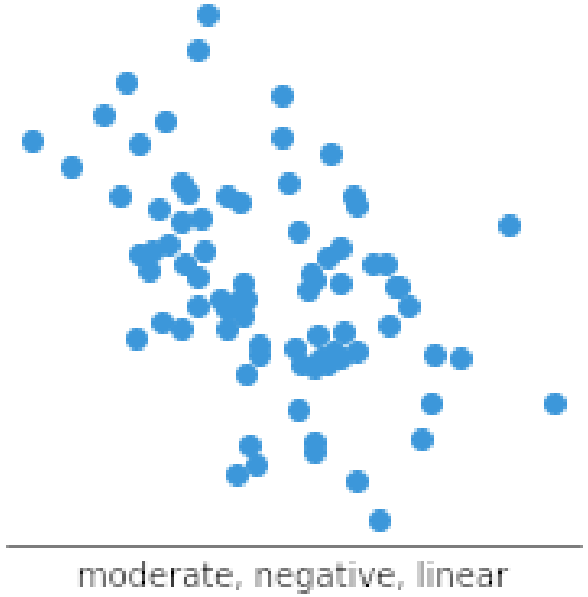
Example: movie genre overlaps



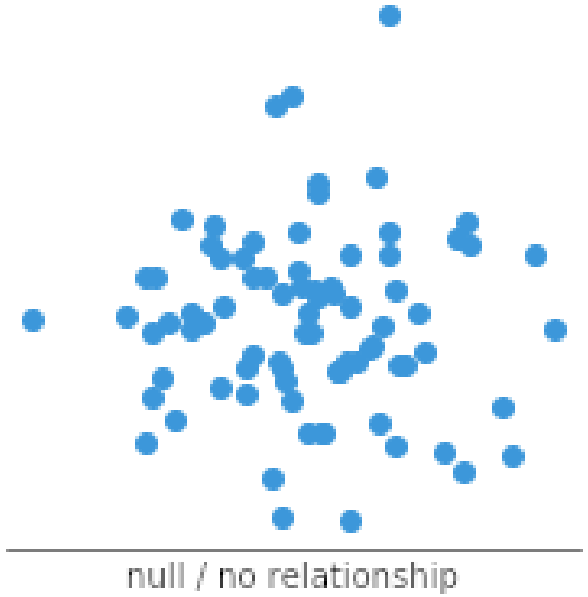
Scatter plots



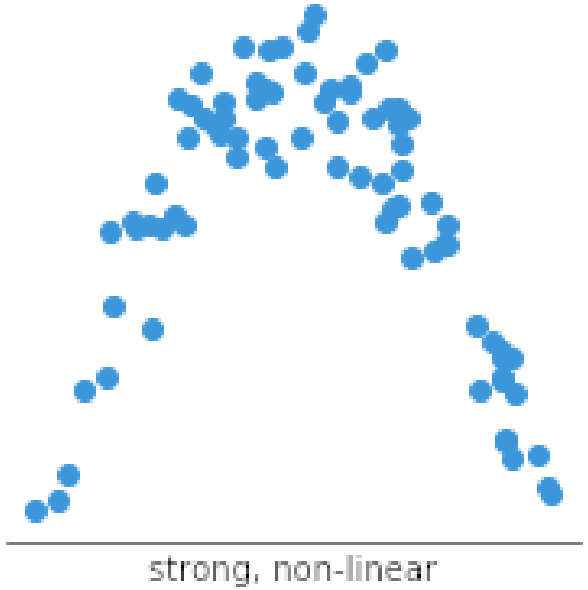
strong, positive, linear



moderate, negative, linear

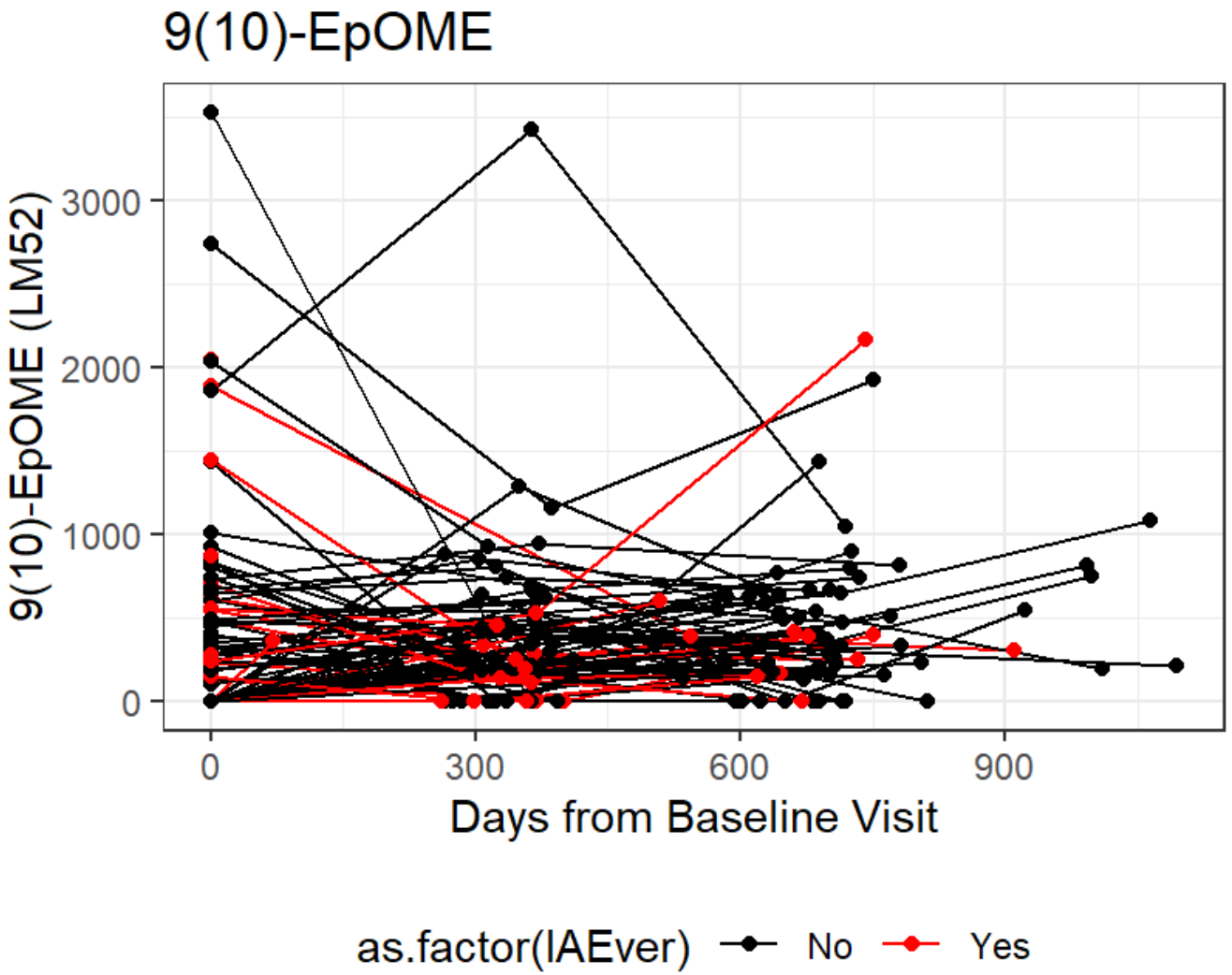


null / no relationship



strong, non-linear

Spaghetti plots



Modifying Variables

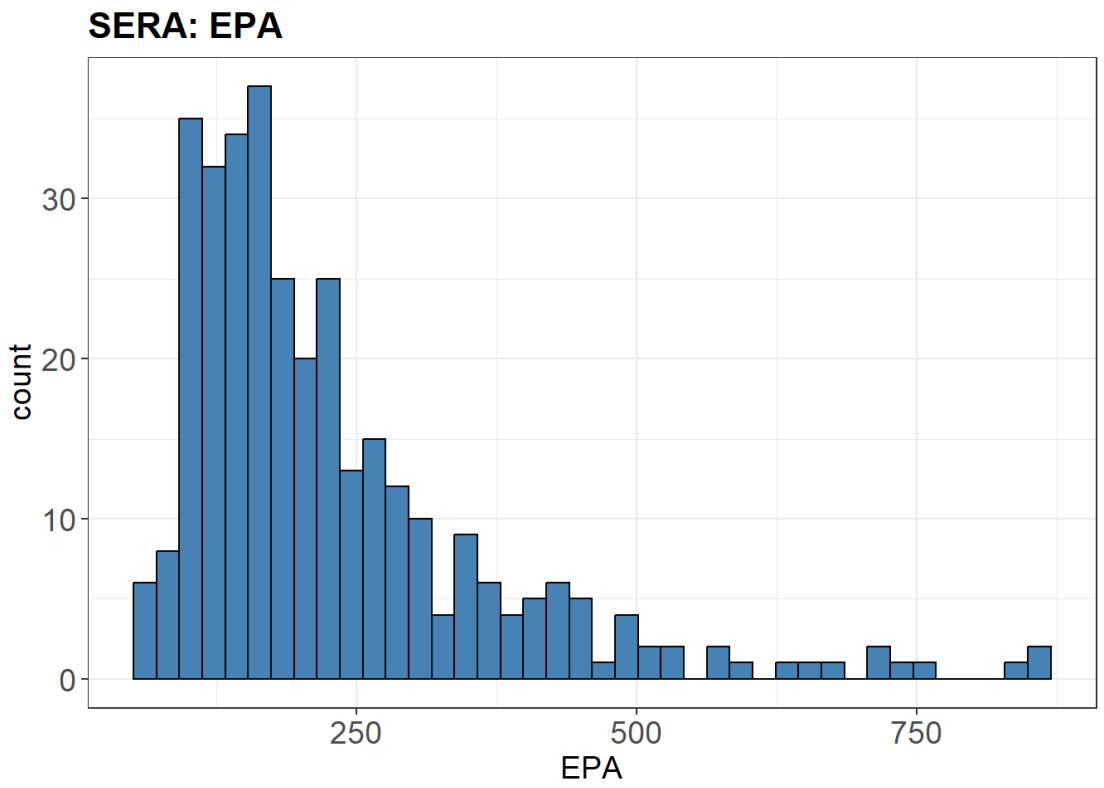
Outliers

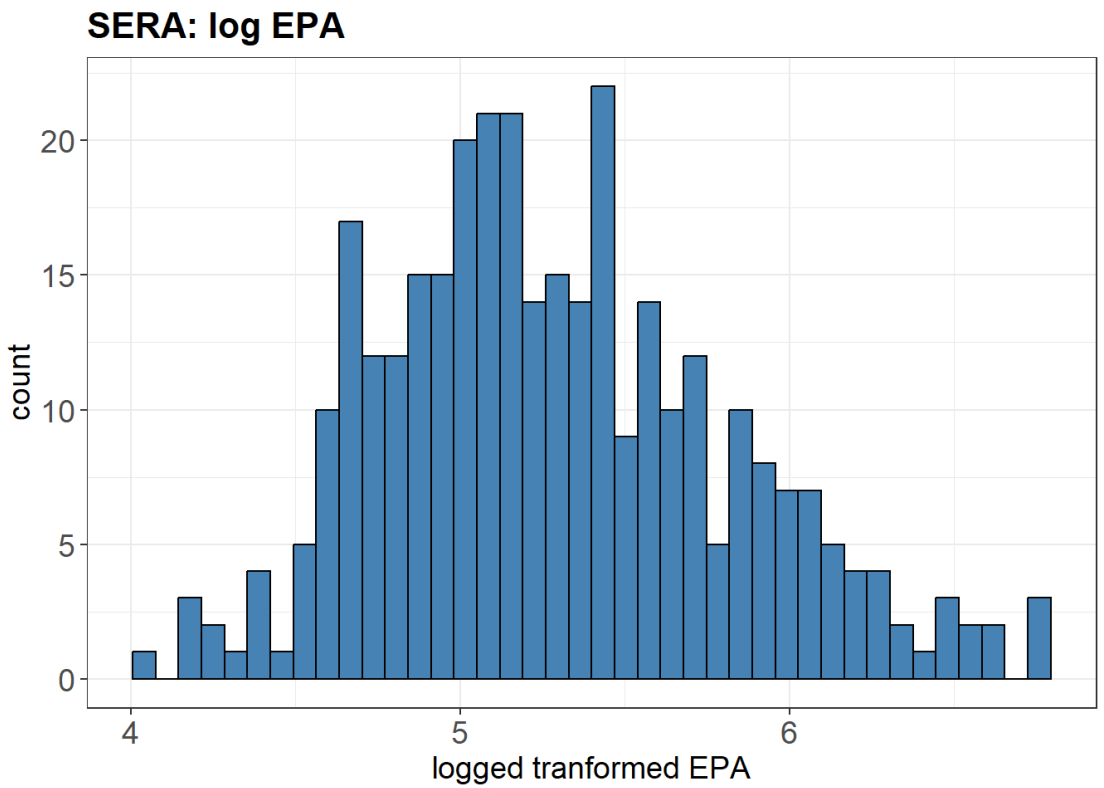
- We can identify outliers in the summary stats and visually
- What is technically considered an “outlier” is really subjective
  - +/- 3SD from the mean
  - “Tukey's method”: points below Q1 - 1.5IQR or above Q3 + 1.5IQR

Solutions

- Transformation: log, box-cox
- Removing
- Change value: winsorization, trimming, or imputation

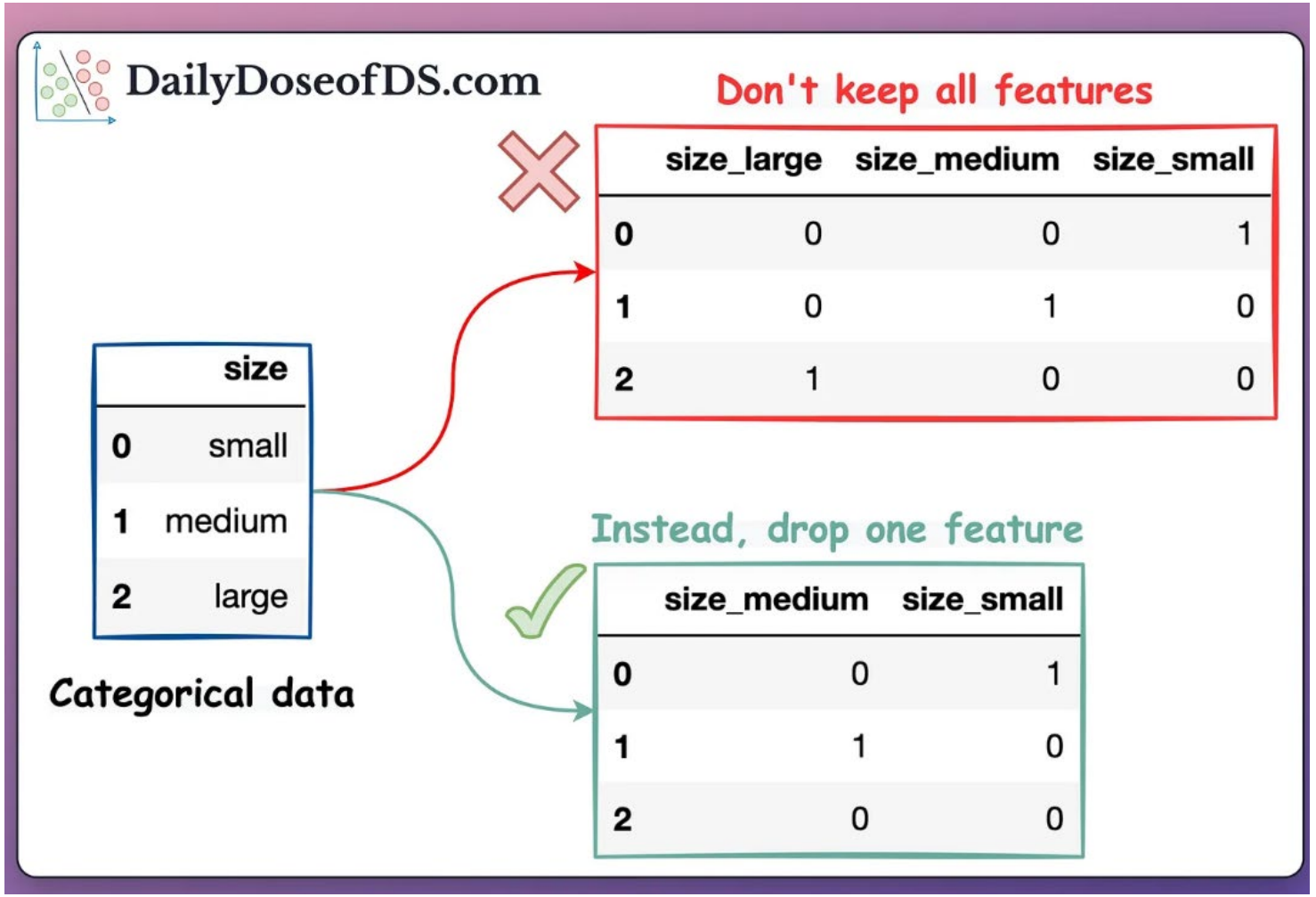
Transformation





Categorical Variables & ML

- Majority of ML only work with numerical variables
- Need to “dummy code” (traditional statistical language) or “one-hot encoding” (more data science language)
- If you keep ALL dummies, induce multicollinearity because you can predict another variable, so remove 1 redundant feature



https://blog.dailydoseofds.com/p/the-most-overlooked-problem-with

Missingness

Machine Learning & Missingness

- Only works on complete data
- Even if each variable contains a small amount of missing data, this can add up in ML methods that use lots of variables and sample size can be reduced DRAMATICALLY

Sample	Variable 1	Variable 2	Variable 3	Variable 4	Variable 5	…	Variable 400,000
Sample 1	Missing						
Sample 2		Missing					
Sample 3					Missing		
…							
Sample 1000							Missing

Why do I have missing data?

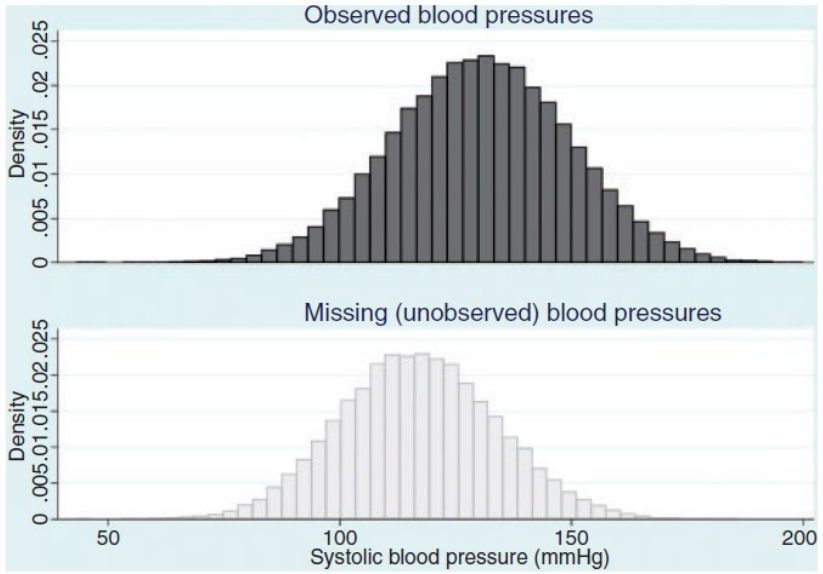
- Survey data:
  - Participant refused to respond or doesn't know the answer
  - Interviewer accidentally skipped questions (or entire sections of instrument)
  - Errors with data entry or quality control
- Biomarker data:
  - Insufficient volume of biospecimen for analysis
  - Undetectable concentration of analyte
  - Laboratory errors, data entry errors
  - Participant refusal to provide biospecimen
- Medical/vital record data:
  - Clinician did not enter data, or data not correctly digitized
  - Lost track of participant due to many different providers/residence

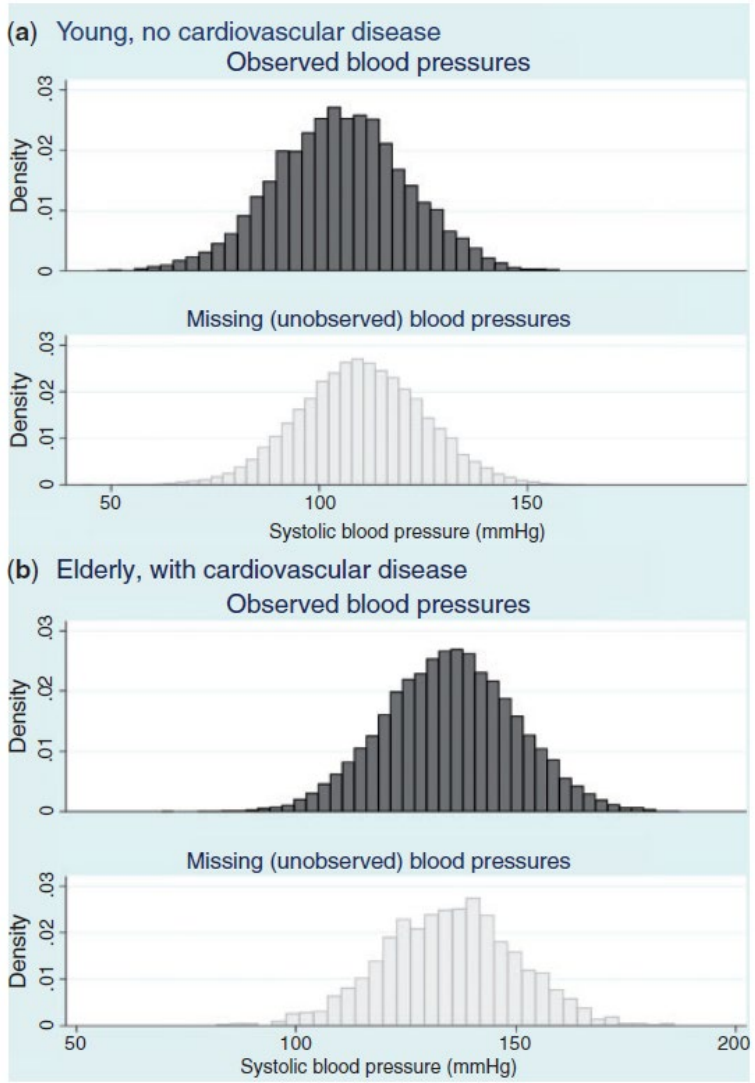
Are participants with missing data different from those with complete data?

Types of missing data

- **Missing completely at random (MCAR):** Missingness of subject's value for Z is independent of every variable in the analysis, including Z
  - Example: When analyzing serum total cholesterol, whether or not the value of cholesterol is missing is unrelated to the actual concentration of cholesterol, and also unrelated to exposure, outcome, covariates (e.g. age, sex, etc.)
- **Missing at random (MAR):** Missingness of subject's value for Z is independent of the value of Z, within levels of (i.e., conditional on) every other variable in the analysis
  - Example: Some participants fail to report smoking behavior. Within categories defined by gender, age, and income, the probability of missingness is not related to whether or not the participant is actually a smoker or not.
- **Missing not at random (MNAR):** Missingness of subject's value for Z is dependent on the actual value of Z (also called non-ignorable)
  - Example: Participants with highest income are least likely to report income.

Missing “at random” is not really random





Source: Bhaskaran and Smeeth (2014) Int J Epidemiol

Common approaches to missing data

- Create a “missing” category and enter a substitute value (e.g. “999”) for participants with missing data
  - Pro: Allows you to use data from all potential participants
  - Con: Biased under many circumstances, including MCAR
- Complete case analysis: Exclude participants from analysis if any missing data for modeled variables
  - Pro: Unbiased under MCAR (and sometimes under MAR)
  - Con: Reduces the number of participants (power), may result in selection bias
- Single imputation: Replace missing entries with a single random draw from distribution (different from imputation with overall sample mean)
  - Pro: Simple and unbiased under MCAR and MAR
  - Con: Underestimates variance, produces incorrect standard errors
- Multiple imputation: Replace missing entries with a set of random draws
  - Pro: Unbiased under MCAR and MAR, appropriately estimates standard errors
  - Con: Relative complexity, require specific analysis packages

How to tell if my data are MCAR, MAR, MNAR

- Create a missing indicator for the variable with missing values. Use bivariate associations (t-test, ANOVA, chi-square) to determine if missingness is associated with levels of other observed variables (MAR) or independent of all observed variables (MCAR).

- Cannot evaluate MNAR vs MAR from available data. May be hypothesized from subject matter knowledge.

Multiple imputation

- Principle: predicting and “filling in” missing values based on the observed associations with other variables in the dataset, and doing this multiple times to achieve better estimate of variability
- Assumption: the probability of missingness depends only on the other observed variables in the dataset (not valid for MNAR)
- Advantages:
  - Allows analysis using the full dataset, unlike complete case analysis
  - Correctly estimates the uncertainty of estimates, by generating multiple possible “fill-in” values for each missing value (unlike single imputation)

Notes on multiple imputation

- As a general rule, you should only impute data if the proportion of missing values is relatively small, typically considered to be less than 5% of your data; if more than 20% of your data is missing, it's usually better to discard that data rather than impute it extensively

- There is randomness to the procedure so results will vary each time. Need to set a **seed** if you want to replicate results.

Does my treatment of missing data matter?

- If less than 5% missing values, probably not
- If 5-15% missing values, maybe
- If >15% missing values, very likely yes

Imputation in Mass Spec Data

- In mass spec data you have missing due to 2 different reasons: values are below LOD (limit of detection) or LOQ (limit of quantitation)
- MissForest used to impute for both types of missingness

sample	7(R)Maresin	RVD5	LTE4	17,18-DiHETE	5,15-DiHETE	LTB4	12,13-DiH	9,10-DiHO	12-HHTrE	14,15-DiH
Sample 01	LOD	LOD	LOD	59.41700244	LOD	LOD	5422.362	10377.43	1746.677	280.2896
Sample 02	LOD	LOD	LOD	52.07075931	LOD	LOD	5381.885	10015.37	LOQ	131.0784
Sample 03	LOD	LOD	LOD	303.8917073	LOD	LOD	2306.883	2745.999	973.6806	165.243
Sample 04	LOD	LOD	LOD	LOD	LOD	LOD	2708.788	3221.369	2728.019	180.7635
Sample 05	LOD	LOD	LOD	LOD	LOD	LOD	1781.521	1487.457	LOQ	115.0951
Sample 06	LOD	LOD	LOD	76.54984434	LOD	LOD	1606.504	2509.821	232.202	112.2789
Sample 07	LOD	LOD	LOD	LOD	LOD	LOD	7292.777	8296.982	234.0174	280.734
Sample 08	LOD	LOD	LOD	122.1791315	LOD	LOD	870.4149	1222.559	LOD	149.2758
Sample 09	LOD	LOD	LOD	61.4169715	LOD	LOD	1783.387	16096.47	LOD	LOQ
Sample 10	LOD	LOD	LOD	402.712427	LOD	LOD	6394.369	8585.591	LOQ	180.4509
Sample 11	LOD	LOD	LOD	93.22550329	LOD	LOD	2015.795	2743.053	276.0801	182.3793
Sample 12	LOD	LOD	LOD	69.64551969	LOD	LOD	1375.769	1743.089	373.7005	151.167
Sample 13	LOD	LOD	LOD	339.0934799	LOD	LOD	1273.772	1257.277	LOD	209.2814
Sample 14	LOD	LOD	LOD	274.9664254	LOD	LOD	17998.58	47105.38	1408.024	355.132
Sample 15	LOD	LOD	LOD	94.86932595	LOD	LOD	3787.201	6140.605	2322.587	214.4578
Sample 16	LOD	LOD	LOD	144.5332806	LOD	LOD	3058.358	5029.876	357.6161	408.5915
Sample 17	LOD	LOD	LOD	LOD	LOD	LOD	3904.006	13297.94	LOQ	173.9718
Sample 18	LOD	LOD	LOD	LOD	LOD	LOD	6204.979	14981.47	577.2835	LOQ
Sample 19	LOD	LOD	LOD	288.0132215	LOD	LOD	5334.095	15567.95	226.0987	279.9562
Sample 20	LOD	LOD	LOD	95.18814788	LOD	LOD	1036.112	1711.13	LOD	195.2592

BIOINFORMATICS ORIGINAL PAPER

Vol. 28 no. 1 2012, pages 112–118
doi:10.1093/bioinformatics/btr597

Data and text mining

Advance Access publication October 28, 2011

MissForest—non-parametric missing value imputation for mixed-type data

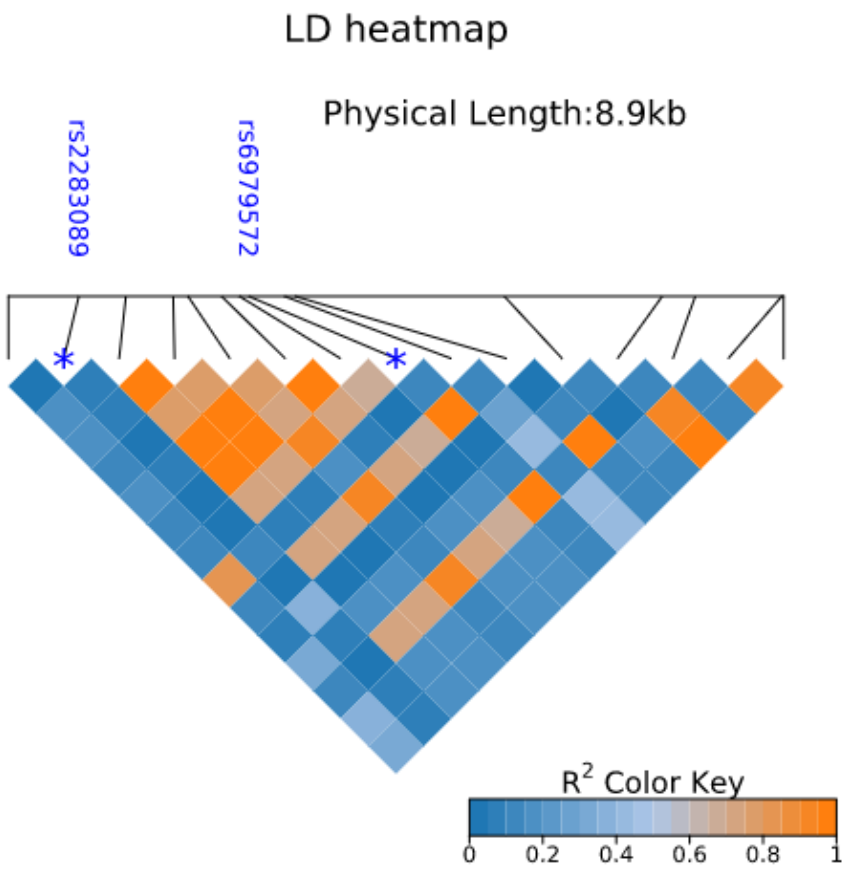
Daniel J. Stekhoven[1,2,3,*] and Peter Bühlmann[1,3]

[1]Seminar for Statistics, Department of Mathematics, ETH Zurich, [2]Life Science Zurich PhD Program on Systems Biology of Complex Diseases and [3]Competence Center for Systems Physiology and Metabolic Diseases, Zurich, Switzerland

Associate Editor: Jonathan Wren

Quick Note on Genetic Imputation

- Many people use impute genome-wide genetic data from array chips
  - i.e. imputing variables we don't have a single data point collected on
- Imput genome-wide set of SNP markers based on reference LD
- Imputation is only as good as your reference
  - Select reference that most aligns with the population you are imputing



Real World Example: NIDDK AI Challenge

NIDDK AI Challenge

https://repository.niddk.nih.gov/data_challenges/data_centric_challenge/

- TEDDY study followed children at-risk for T1D (FDR or high risk HLA allele) to understand the environmental factors that contribute to the disease
- Across multiple sites in USA and Europe
- Assays performed at different sites (example: CO got grant to assay DNAm, FL RNA-seq data…)
- Winners got free access to the AI ready data!

About the Competition

For the NIDDK-CR Data Centric Challenge, NIDDK sought innovative approaches to enhance the utility of NIDDK datasets for AI applications. Towards this, the goals of the competition were to 1) generate an "AI-ready" dataset that can be used for future data challenges, and 2) to produce methods that can be used to enhance the AI-readiness of NIDDK data. Participants enhanced de-identified data from the following longitudinal studies focused on Type 1 Diabetes (T1D) that are available through NIDDK-CR:

- The Environmental Determinants of Diabetes in the Young (TEDDY) study
- Four studies from the Type 1 Diabetes TrialNet (TrialNet) network, including TN01, TN16, TN19, and TN20 (see Intermediate/Advanced-level participation description below for further details)

Participation in this challenge was tiered based on the challenge applicants' self-described experience with data science and analytics (i.e., beginner or intermediate/advanced). Participants were instructed to 1) prepare a single dataset by aggregating all data files associated with one or more longitudinal studies on T1D listed above, and 2) augment the single dataset to ensure AI-readiness. One winner from each group below was selected.

Beginner-Level Challenge: The goal for challenge participants was to aggregate the 48+ datasets from the TEDDY study into a single unified and machine-readable dataset harmonized by participant ID (MaskID). Since NIDDK cannot know what other study designs may arise in the future, or what discoveries could be pursued when combining the TEDDY dataset with other datasets, AI-readiness required aggregation of all 48 dataset files into a single tabular (i.e., spreadsheet or rectangular) .csv file type; data enhancement steps that do not meaningfully alter the original data; and preparation of dataset documentation that is both human- and machine-readable.

Team Members

Milton Pividori

Marc Subirana Granés

Randi Johnson

Sarah Slack

Kirk Hohsfield

Lauren Vanderlinden

Reading in Data

- 48 datasets is a LOT
- Noticed capitalization was just a mess between datasets
- Sometimes “MaskID”, “maskid”, “MASKID”

```r
# Script for R functions used in raw data merge.


# Read in all data files, with dataframe name as file name
# Set all dataframe names and column names to lowercase
read_data <- function(file) {
    name <- tolower(basename(gsub("\\.csv", "", file, ignore.case = T)))
    # print(name)
    assign(name, read.csv(file, na.strings=c("","  ", "NA", NA))
           %>% set_names(., tolower(names(.))))
}
```

Check if longitudinal data

- We know some datasets collected longitudinal data, some didn't
- Really need to know what variable to merge on the cross-sectional datasets did NOT include time variable

```
# Check if data is longitudinal (returns TRUE if longitudinal)
check_long <- function(df) {
    dist_ids <- n_distinct(df$maskid)
    rows <- nrow(df)
    return(dist_ids != rows)
}
```

- For example, some cross-sectional datasets included "at-birth survey", "9 month survey"

What type of identifier is included?

- Maskid was suppose to be the individual id in all datasets
- However, in longitudinal datasets there may be different sample id variable names (it was messy…)
- This one required lots of “institutional knowledge” of individuals working with TEDDY data before

```r
# Check which "identifiers" are present in dataset
check_ids <- function(df, cols_to_count) {
    counts <- rep(0, length(cols_to_count))

    for (i in seq_along(cols_to_count)) {
        col <- cols_to_count[i]
        if (col %in% colnames(df)) {
            counts[i] <- 1
        }
    }
    result <- setNames(counts, cols_to_count)
    return(result)
}
```

Merge multiple datasets

Cross-sectional merge

```r
# Merge non-longitudinal dataframes by MaskID
merge_dfs <- function(list_of_dfs) {
    merge_two_dfs <- function(df1, df2) {
        merge <- full_join(df1, df2, by = "maskid")
        return(merge)
    }
    merge_result <- Reduce(merge_two_dfs, list_of_dfs)
    return(merge_result)
}
```

Longitudinal merge

```r
# Function to merge list of dataframes by given merge columns
# To note: all merge_cols values can be converted to integer
merge_dfs_long <- function(list_of_dfs, merge_cols) {
    # Function to merge two dfs
    merge_two_dfs <- function(df1, df2) {
        # Set non-MaskID merge columns to character
        df1 <- df1 %>% mutate_at(vars(all_of(merge_cols)), as.integer)
        df2 <- df2 %>% mutate_at(vars(all_of(merge_cols)), as.integer)

        # Merge
        merge <- full_join(df1, df2, by = merge_cols, relationship = "many-to-many")
        return(merge)
    }
    # Combined merge
    merge_result <- Reduce(merge_two_dfs, list_of_dfs)
    return(merge_result)
}
```

Check bad variables

- Any column missing all data was removed
- We identified 1 dataset (particular assay) that truly failed and was missing vast majority of their variables. We ended up just removing that dataset entirely

```r
# Check how many dataframe columns are completely NA
check_na <- function(df) {
    count <- 0
    for (col in colnames(df)) {
        if (sum(is.na(df[[col]])) == nrow(df)) {
            count <- count + 1
        }
    }
    return(count)
}
```

Longitudinal “Time”

```r
## For datasets with only eventAge, let's calculate a due num to merge with other datasets that have due num but NOT event age
getDueNum <- function(eventAge){
  ageYears = eventAge/365.25
  ageMonths = ageYears*12
  dueNum_calc <- round(ageMonths/3) * 3
  return(dueNum_calc)
}
```

- TEDDY had set follow-up check points (every 6 months for first 3 years and then annually)
- Some longitudinal datasets had “age” and some had “due num” that reflected the visit # that they are suppose to be checked up on

Flagging potential errors

```
### QC on Longitudinal Data: This is checking if we have conflicting info on longitudinal data.  We would expect things to be updated (like going from NA to a value)
flag_missingInfo_mostRecentVisit <- function(ID, dataset){
    tmp <- dataset[which(dataset$maskid == ID),]
    vars <- colnames(tmp)
    if("event_age" %in% vars){tmp <- tmp[order(tmp$event_age, decreasing=TRUE),]}else{tmp <- tmp[order(tmp$visit, decreasing=TRUE),]}

    flag=rep(0, ncol(tmp))
    names(flag) = colnames(tmp)

    if(nrow(tmp)==1){flag=flag}else{
    for(c in 1:ncol(tmp)){
     for(r in 2:nrow(tmp)){
         if(identical(tmp[(r-1),c], tmp[r,c])|is.na(tmp[r,c])){flag[c]=flag[c]}else{flag[c]=flag[c]+1}
     }
    }
   }
   # return(flag)
   flag2 = flag[-which(names(flag) %in% c("event_age", "visit"))]
   return(flag2)
}
```

Collapsing down longitudinal

- We made the decision most ML doesn't account for repeated measures, so making data wide and using time as a different variable:
  - Example

Long format

ID	Visit	Gene A
1	1	100
1	2	120
2	1	87
2	2	65
…		

Wide format

ID	Gene A at Visit 1	Gene A at Visit 2
1	100	120
2	87	65
…		

```
collapseLong <- function(ID, dataset){
  #get a tmp variable for the ID and order with most recent visit at top;
  tmp <- dataset[which(dataset$maskid == ID),]
  vars <- colnames(tmp)
  if("event_age" %in% vars){
      tmp <- tmp[order(tmp$event_age, decreasing=TRUE),]
      ageVar = "event_age"
  }else if("effective_age" %in% vars){
      tmp <- tmp[order(tmp$effective_age, decreasing=TRUE),]
      ageVar = "effective_age"
  }else{
      tmp <- tmp[order(tmp$visit, decreasing=TRUE),]
      ageVar = "visit"}

  ### identify duplicate IDs/age combintaiton and make a new tmp2 matrix without these duplicates ###
  if(nrow(tmp) == length(unique(tmp[,ageVar]))){tmp2 = tmp}else{
      tmp2 = matrix(nrow = length(unique(tmp[,ageVar])), ncol = ncol(tmp))
    colnames(tmp2) = colnames(tmp)}

  if(nrow(tmp2) != nrow(tmp)){
    for(c in 1:ncol(tmp)){
```

```
119         for(c in 1:ncol(tmp)){
120           for(r in 2:nrow(tmp)){
121             # not duplicate, move forward with value
122             if(tmp[r-1,ageVar] != tmp[r,ageVar]){tmp2[r-1, c] = tmp[r-1,c]}
123             #if one duplicates, but have same value, just take the value
124             else if(tmp[(r-1), ageVar]==tmp[r,ageVar] & identical(tmp[(r-1), c], tmp[r,c])){tmp2[r-1,c]=tmp[r-1,c]}
125           #if duplicates and one value is NA, take the other
126               else if(is.na(tmp[r-1,c])){tmp2[r-1,c]=tmp[r,c]}
127               else if(is.na(tmp[r, c])){tmp2[r-1,c]=tmp[r-1,c]}
128           #if the discordinate values AND numeric, take the one closest to the median
129             else if(is.numeric(dataset[,c])){
130               median_var = median(dataset[,c], na.rm=TRUE)
131               vals = c(tmp[r-1,c], tmp[r,c])
132               r_1_distance = abs(median_var-tmp[r-1,c])
133               r_distance = abs(median_var-tmp[r,c])
134               val_want = vals[which.min(c(r_1_distance, r_distance))]
135               tmp2[r-1,c] = val_want}
136             #if discordinate values are not numeric, just take top value;
137             else{tmp2[r-1,c] = tmp[r-1,c]}
138           }}}
```

```
139
140       tmp2 = as.data.frame(tmp2)
141
142     ### Collapse Longitudinal Down ###
143
144       # set up a vector that we want to output;
145       want = matrix(nrow=1, ncol=ncol(tmp2))
146       colnames(want) = colnames(tmp2)
147
148       for(c in 1:ncol(tmp2)){
149         want[1,c] = first(na.omit(tmp2[,c]))
150       }
151
152       return(as.data.frame(want))
153     }
154
```

Address Outliers

```r
removeOutliers <- function(dataset){
  dat.noOuts <- dataset
  numericVars <- colnames(select_if(dataset, is.numeric))
  vars <- numericVars[-which(numericVars %in% c("maskid", "event_age", "visit"))]
  nUniqueVals <- c()
  for(c in vars){
    nUniqueVals <- c(nUniqueVals, length(unique(dataset[,c])))
  }
  vars2 <- vars[which(nUniqueVals>2)]
  for(c in vars2){
    median_var = median(dataset[,c], na.rm=TRUE)
    sd_var = sd(dataset[,c], na.rm=TRUE)
    for(r in 1:nrow(dat.noOuts)){
      if(is.na(dat.noOuts[r,c]) | dat.noOuts[r,c] > median_var+5*sd_var | dat.noOuts[r,c]< median_var-5*sd_var){dat.noOuts[r,c]=NA}else{dat.noOuts[r,c] = dat.noOuts[r,c]}
    }
  }
  return(dat.noOuts)
}
```

- We called outliers as anything outside 5 standard deviations of the median
- We removed those values and replaced with missing

Remove Repeated Variables

```
238   ### Get indicator variables for those multiple similar columns we have ###
239   condenseRepeatedVars <- function(dataset, vars_all){
240     #find the variables that are repeated names and only difference is numbers in name
241     #vars_all <- colnames(dataset)
242     vars_noDupNum <- gsub("[0-9]", "", vars_all)
243     varTab <- as.data.frame(table(vars_noDupNum))
244     vars_toCondense <- varTab[which(varTab$Freq>1), 1]
245
246     #make a new dataset, first by defining it as the dataset with only unique variables that don't need condensing
247     dataset2 <- dataset[,-which(vars_noDupNum %in% vars_toCondense)]
248     #now for the variables that need to be condensed, make new variable(s) to add onto dataset2
249     for(i in vars_toCondense){
250       tmp <- dataset[,which(vars_noDupNum == i)]
251       responses <- c()
252       for(j in 1:ncol(tmp)){
253         responses <- c(responses, tmp[,j])}
254       unique_responses = unique(responses[!is.na(responses)])
255
256       toAdd <- matrix(0, nrow=nrow(tmp), ncol = length(unique_responses))
```

```
256         toAdd <- matrix(0, nrow=nrow(tmp), ncol = length(unique_responses))
257         for(c in 1:ncol(toAdd)){
258           for(r in 1:nrow(toAdd)){
259             toAdd[r,c] <-  ifelse(sum(grepl(unique_responses[c], tmp[r,]))>0, 1, 0)
260           }}
261         colnames(toAdd) <- paste0(i, "_response_", unique_responses)
262 
263           if(i == vars_toCondense[1]){toAdd_final = toAdd}else{toAdd_final = cbind(toAdd_final, toAdd)}
264       }
265         dataset3 = cbind(dataset2, toAdd_final)
266       return(dataset3)}
```

Removing low frequency variables

```r
### To remove the low frequency variables ###
findLowFreqVars <- function(var){
    toRm = ifelse(length(table(var))==2 & sum(names(table(var)) %in% c("0", "1"))==2 & min(table(var)/length(var))<0.05, 1, 0)
    return(toRm)
}

#apply this over every variables in function which will;
removeLowFreqVars <- function(dataset){
  findLow = apply(dataset, 2, function(a) findLowFreqVars(a))
  toRm = names(findLow[which(findLow==1)])
  dat2 <- dataset[,-which(colnames(dataset) %in% toRm)]
  return(dat2)
}
#example:
#dat_clean <- removeLowFreqVars(dat)
```

Python code for one-hot encoding

```
import numpy as np
import pandas as pd
import sklearn.preprocessing


# Read in the CSV file
dataFrame = pd.read_csv('/home/ec2-user/SageMaker/studies/TEDDY/TEDDY_Data/TEST_RESULTS.csv')
print(dataFrame)
print(dataFrame.describe())
print(dataFrame.dtypes)

# Pivot the DataFrame
dataFrame = dataFrame.pivot(index='MaskID' , columns='TEST_NAME', values='RESULT')
print(dataFrame)
print(dataFrame.describe())
print(dataFrame.dtypes)

# Encoding labels
categoricalColumnNames = dataFrame.select_dtypes(include=('object')).columns.values.tolist()
print(categoricalColumnNames)
len(categoricalColumnNames)
```

```
for columnName in categoricalColumnNames:

    # Encoding column
    encoder = sklearn.preprocessing.OneHotEncoder(handle_unknown='ignore', sparse_output=False)
    encoder.fit(dataFrame[columnName].to_numpy().reshape(-1,1))
    data = encoder.transform(dataFrame[columnName].to_numpy().reshape(-1,1))

    # Copying to data frame (ignoring non-string categories)
    for i in range(data.shape[1]):
        if type(encoder.categories_[0][i]) == str:
            dataFrame[columnName + '_' + encoder.categories_[0][i]] = data[:, i]
            print(columnName + '_' + encoder.categories_[0][i])

    dataFrame.pop(columnName)
    columns = [c for c in dataFrame.columns.values.tolist() if columnName in c]
    mask = dataFrame[columns].sum(axis=1).values == 0
    dataFrame.loc[mask, columns] = np.nan

print(dataFrame)
```

Repositories

- The proposal got a perfect score! Just did not pull off the overall win.
- We could only access data on the NIDDK workbench and had issues with environments
- Also first time we worked together on a project with all of us working on code at the same time, leading to git issues.
- Our teams code:

https://github.com/pivlab/niddk-ai-challenge/tree/main

- Winner's code: https://github.com/niddk-data-challenge/Beginner-Level-Challenge-AI-Ready-TEDDY-Dataset

Conclusions

- No standard on how to best pre-process data
- Examine quality both visually and with summary stats
- Missing data is an issue in ML
- Examining and pre-processing data can take just as long or even longer than running the statistical analyses itself