HEADS

PhD PROGRAMME IN HEALTH DATA SCIENCE

Data Management

Basics of Health Intelligent Data Analysis

PhD Programme in Health Data Science

Cláudia Camila Dias

Pedro Pereira Rodrigues

U.PORTO
FMUP FACULDADE DE MEDICINA UNIVERSIDADE DO PORTO

MEDCIDS
DEPARTAMENTO DE MEDICINA DA COMUNIDADE, INFORMAÇÃO E DECISÃO EM SAÚDE

CINTESIS
Health. Research.

Exercises

- Open the Dataset_Obst_Example1.csv with clinical data about new born.

- Checks that:

  - the database has been correctly imported.

  - the variable type fits the variable in question.

  - the missings were well defined.

Exercises

- How many newborns the database has.
- Which variables are recorded in the database

str()

- ***str(object, …)*** <- compactly display the structure of an R object

```
>str(data)
'data.frame':  250 obs. of  10 variables:
 $ SubjectID          : Factor w/ 250 levels "Subject001","Subject002",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ Mother_Age         : int  31 25 33 37 27 28 31 32 19 28 ...
 $ Mother_Weight      : int  55 66 58 53 56 58 59 68 84 58 ...
 $ Mother_Height      : int  165 160 172 172 162 158 157 162 159 171 ...
 $ Newborn_Gender     : int  0 1 0 1 1 0 0 1 1 0 ...
 $ Delivery_Methods   : Factor w/ 3 levels "Cesarian","Forceps",..: 2 3 2 3 3 2 3 3 3 3 ...
 $ Beginning_of_labour: Factor w/ 2 levels "Induced","Spontaneous": 1 1 2 2 1 1 2 2 1 2 ...
 $ Gestational_Age    : int  39 41 40 38 40 40 40 36 40 39 ...
 $ Newborn_Weight     : int  2805 3525 3830 3395 3290 3175 3215 965 2835 2860 ...
 $ Newborn_Height     : int  48 52 50 51 51 50 48 33 51 48 ...
```

summary

- ***summary(object, …)*** <- produce result summaries of the results of various model fitting functions.

```
> summary(data)
  SubjectID     Mother_Age    Mother_Weight    Mother_Height   Newborn_Gender
 Subject001:  1   Min.   :15.00   Min.   :  0.00   Min.   :144.0   Min.   :0.000
 Subject002:  1   1st Qu.:27.00   1st Qu.: 55.00   1st Qu.:157.0   1st Qu.:0.000
 Subject003:  1   Median :31.00   Median : 61.00   Median :162.0   Median :0.000
 Subject004:  1   Mean   :30.33   Mean   : 62.01   Mean   :162.1   Mean   :0.496
 Subject005:  1   3rd Qu.:34.00   3rd Qu.: 69.75   3rd Qu.:166.0   3rd Qu.:1.000
 Subject006:  1   Max.   :46.00   Max.   :130.00   Max.   :193.0   Max.   :1.000
 (Other)   :244
 Delivery_Methods  Beginning_of_labour Gestational_Age Newborn_Weight Newborn_Height
 Cesarian: 68      Induced    : 85      Min.   :23.00   Min.   : 390   Min.   :27.00
 Forceps : 63      Spontaneous:158      1st Qu.:38.00   1st Qu.:2801   1st Qu.:47.00
 Vaginal :113      NA's       :  7      Median :39.00   Median :3132   Median :49.00
 NA's    :  6                          Mean   :38.62   Mean   :3067   Mean   :48.51
                                       3rd Qu.:40.00   3rd Qu.:3410   3rd Qu.:50.00
                                       Max.   :41.00   Max.   :4410   Max.   :54.00
```

Exercises

- Considers that in the gender variable, 1 corresponds to Male and 0 to Female

- Transform the variable `Newborn_Gender` into a factor

- In the variable *Beginning_of_labour* considers Spontaneous as the first category

Exercises

- Consider the newborns whose mother was over 40 years old.
  - How many are there?
  - Which ones?
- What is the gestational age of these newborns?

which()

- ***which(x, arr.ind = FALSE, useNames = TRUE)***

```
> which(data$Mother_Age>40)
```

```
> length(which(data[,”Mother_Age”]>40))
```

```
> data[data$Mother_Age>40,"Gestational_Age"]
```

Exercises

- How many induced and spontaneous births are there?

- How many of the newborns from each type of delivery were born by C-section or vaginal delivery?

table()

- ***table()*** <- build a contingency table of the counts at each combination of factor levels.

```
> table(data$Beginning_of_labour)

    Induced Spontaneous
         85         158

>
table(data$Beginning_of_labour,data$Delivery_Methods)

             Cesarian Forceps Vaginal
  Induced           17      29      38

  Spontaneous       50      33      73
```

- ***prop.table(x, margin = NULL)*** <- express table entries as fraction of marginal table. if margin has length zero, then one gets x/sum(x).

```
> prop.table(table(data$Beginning_of_labour))

    Induced Spontaneous
  0.3497942   0.6502058

>
prop.table(table(data$Beginning_of_labour,data$Delivery_Methods
))

                Cesarian    Forceps    Vaginal
  Induced     0.07083333 0.12083333 0.15833333
  Spontaneous 0.20833333 0.13750000 0.30416667
```

More functions - Character

- ***substr(x, start, stop)***
- ***substring(text, first, last = 1000000L)***

extract or replace substrings in a character vector.

```
> data$ID <-
substr(data$SubjectID,8,10)
```

- ***strsplit(x, split, fixed = FALSE,…)*** <- split the elements of a character vector

```
>strsplit("Winter School"," ")
[[1]]
[1] "Winter" "School”
```

```
>strsplit(”file.csv",
".”,fixed=T)
[[1]]
[1] "file" "csv"
```

Exercises

- Considers all babies born at less than 36 weeks to be premature.
- Creates a new variable that classifies if the newborn is premature or not.
- Creates a new date frame with term babies with only the information referring to the mother.

subset

- ***subset(x, subset, select,…)*** <- return subsets of vectors, matrices or data frames which meet conditions.

```
> data_Termo <- subset(x=data,subset=data$Prematuro==0,select = 1:4)
> head(data_Termo)
   SubjectID Mother_Age Mother_Weight Mother_Height
1 Subject001         31            55           165
2 Subject002         25            66           160
3 Subject003         33            58           172
4 Subject004         37            53           172
5 Subject005         27            56           162
6 Subject006         28            58           158
```

subset

- ***subset(x, subset, select,…)*** <- return subsets of vectors, matrices or data frames which meet conditions.

```
> data_Termo2 <- subset(x=data,subset=(data$Prematuro==0 & data$Beginning_of_labour=="Induced"),select =
c("SubjectID","Newborn_Weight"))

> head(data_Termo2)

    SubjectID Newborn_Weight

1  Subject001           2805

2  Subject002           3525

5  Subject005           3290

6  Subject006           3175

9  Subject009           2835

11 Subject011           3095
```

Exercises

- What is the average weight in premature newborn? And in non-premature newborn?

by()

- ***by(data, INDICES, FUN, …, simplify = TRUE)*** <- is an object-oriented wrapper for ***tapply*** applied to data frames.

```
> by(data,data$Prematuro,FUN = function(x){
+   mean.age <- mean(data$Newborn_Weight)})
data$Prematuro: 0
[1] 3066.54
------------------------------------------------------------------------
data$Prematuro: 1
[1] 3066.54
```

Exercises

- Open the *Dataset_Obst_Example2.csv* database with clinical CTG information of newborns.

%in%

- ***%in%***

returns a logical vector indicating if there is a match or not for its left operand

```
> y <- 5:-2
[1]  5  4  3  2  1  0 -1 -2
> 6 %in% y

> 5 %in% y

> c(6,5,4) %in% y

>y1<- sample(y)
```

Match and identical

- ***match(x, table, nomatch = NA_integer_, incomparables = NULL)***

returns a vector of the positions of (first) matches of its first argument in its second

```
> match(data$SubjectID,dataCTG$SubjectID)
```

- ***identical(x, y,…)*** <- test two objects for being ***exactly*** equal

```
> identical(data$SubjectID,dataCTG$SubjectID)
```

```
> identical(data$SubjectID, data$SubjectID)
```

More Functions - Merge

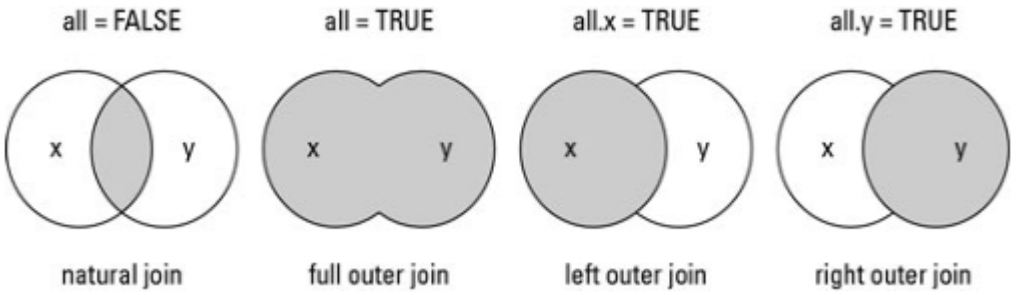
- ***merge(x, y, by = intersect(names(x), names(y)), by.x = by, by.y = by, all = FALSE, all.x = all, all.y = all, sort = TRUE, suffixes = c(".x",".y"), no.dups = TRUE, incomparables = NULL, …)*** <- merge two data frames by common columns or row names

- **by, by.x, by.y:** The names of the columns that are common to both x and y. The default is to use the columns with common names between the two data frames.

- **all, all.x, all.y:** Logical values that specify the type of merge. The default value is all=FALSE(meaning that only the matching rows are returned).

More Functions - Merge

- **Natural join:** To keep only rows that match from the data frames, specify the argument all=FALSE.
- **Full outer join:** To keep all rows from both data frames, specify all=TRUE.
- **Left outer join:** To include all the rows of your data frame x and only those from y that match, specify all.x=TRUE.
- **Right outer join:** To include all the rows of your data frame y and only those from x that match, specify all.y=TRUE.



More Functions - Merge

```
> dataAll <- merge(data, dataCTG,by = "SubjectID",all = T)

> dataX <- merge(data, dataCTG,by = "SubjectID",all.x = T)

> dataY <- merge(data, dataCTG,by = "SubjectID",all.y = T)

> dataInt <- merge(data, dataCTG,by = "SubjectID",all = F)
```

Exercises

- Save in a new file a new data frame with the natural merge join.

Apply Functions

- ***apply(X, MARGIN, FUN, …)*** <- returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

- ! apply coerces everything into the same type. Numbers can become characters !

- ***lapply(X, FUN, …)***
- ***lapply*** <- returns a list of the same length as X, each element of which is the result of applying FUN to the corresponding element of X.

- ***sapply*** <- is a wrapper of lapply by default returning a vector, matrix or array

- ***vapply*** <- is similar to sapply, but has a pre-specified type of return value

Apply Functions

- ***apply(X, MARGIN, FUN, …)*** <- returns a vector or array or list of values obtained by applying a function to margins of an array or matrix.

- ***MARGIN –*** for a matrix 1 indicates rows, 2 indicates columns, c(1, 2) indicates rows and columns

- ! apply coerces everything into the same type. Numbers can become characters !

```
> apply(data[,2:4],2,mean)
   Mother_Age Mother_Weight Mother_Height
       30.328        62.012       162.104
```

Apply Functions

```
> i <- sapply(data,is.numeric)
> i
  SubjectID          Mother_Age       Mother_Weight       Mother_Height
              FALSE                TRUE                TRUE                TRUE
     Newborn_Gender    Delivery_Methods Beginning_of_labour     Gestational_Age
                TRUE               FALSE               FALSE                TRUE
     Newborn_Weight      Newborn_Height           Prematuro
                TRUE                TRUE                TRUE
> sapply(data[,i],mean,na.rm = T)
 Mother_Age   Mother_Weight   Mother_Height  Newborn_Gender Gestational_Age
         30.328          62.012         162.104           0.496          38.620
 Newborn_Weight  Newborn_Height       Prematuro
       3066.540          48.508           0.096

> lapply(data[,i],mean,na.rm=T)
```

Exercises

- Open the database *Dataset_Diab_Example1_long.csv* with information about diabetic patients.

Exercises

- Open the database *Dataset_Diab_Example1_long.csv* with information about diabetic patients.

```
> data_D <- read.csv("Dataset_Diab_Example1_long.csv")
> head(data_D)
  Subject Gender    Group Age Medication Measure Glucose
1       1      F Diabetes  65   acarbose       1     185
2       2      M Diabetes  53   miglitol       1     182
3       3      M Diabetes  60   acarbose       1     180
4       4      M Diabetes  60   acarbose       1     170
5       5      F Diabetes  55   miglitol       1     166
6       6      M Diabetes  70   miglitol       1     162
```

Wide and long formats

- ***reshape(data, varying = NULL, v.names = NULL, timevar = "time", idvar = "id", ids = 1:NROW(data), times = seq_along(varying[[1]]), drop = NULL, direction, new.row.names = NULL, sep = ".", split = if (sep == "") { list(regexp = "[A-Za-z][0-9]", include = TRUE) } else { list(regexp = sep, include = FALSE, fixed = TRUE)})***

reshapes a data frame between 'wide' format with repeated measurements in separate columns of the same record and 'long' format with the repeated measurements in separate records.

Wide and long formats

- ***reshape(data, varying = NULL, v.names = NULL, timevar = "time", idvar = "id", ids = 1:NROW(data), times = seq_along(varying[[1]]), drop = NULL, direction, new.row.names = NULL, sep = ".", split = if (sep == "") { list(regexp = "[A-Za-z][0-9]", include = TRUE) } else { list(regexp = sep, include = FALSE, fixed = TRUE)})***

reshapes a data frame between 'wide' format with repeated measurements in separate columns of the same record and 'long' format with the repeated measurements in separate records.

```
### Convert to wide

> dataD_w <- reshape(dataD, idvar="Subject",
v.names = "Glucose",timevar="Measure",
direction="wide")

> head(dataD_w,n=2)

 Subject Gender    Group Age Medication

1       1      F Diabetes  65   acarbose

2       2      M Diabetes  53   miglitol

  Glucose.1 Glucose.2 Glucose.3 Glucose.4

1       185       219       181       123

2       182       220       177       122
```

Wide and long formats

- ***reshape(data, varying = NULL, v.names = NULL, timevar = "time", idvar = "id", ids = 1:NROW(data), times = seq_along(varying[[1]]), drop = NULL, direction, new.row.names = NULL, sep = ".", split = if (sep == "") { list(regexp = "[A-Za-z][0-9]", include = TRUE) } else { list(regexp = sep, include = FALSE, fixed = TRUE)})***

reshapes a data frame between 'wide' format with repeated measurements in separate columns of the same record and 'long' format with the repeated measurements in separate records.

```
### Convert to long

> dataD_long <- reshape(dataD_w, idvar="Subject", varying=list(c("FirstM","SecondM","ThirdM","FourthM")), v.names="Glucose", timevar="Measure", direction="long")

>

> head(dataD_long,n=2)

Subject Gender    Group Age Medication
1.1       1       F Diabetes  65   acarbose
2.1       2       M Diabetes  53   miglitol
3.1       3       M Diabetes  60   acarbose
    Measure Glucose
1.1       1     185
2.1       1     182
3.1       1     180
```

Order and sort

```
> sort(dataD_long$Subject)
```

```
> order(dataD_long$Subject)
```

```
> > dataD_long <-dataD_long[order(dataD_long$Subject,dataD_long$Glucose),]
> head(dataD_long)
```

	Subject	Gender	Group	Age	Medication	Measure	Glucose
1.4	1	F	Diabetes	65	acarbose	4	123
1.3	1	F	Diabetes	65	acarbose	3	181
1.1	1	F	Diabetes	65	acarbose	1	185
1.2	1	F	Diabetes	65	acarbose	2	219
2.4	2	M	Diabetes	53	miglitol	4	122
2.3	2	M	Diabetes	53	miglitol	3	177

aggregate

- ***aggregate()*** <- splits the data into subsets, computes summary statistics for each, and returns the result in a convenient form.

```
> data_Ag <- aggregate(data_D,by = list(data_D$Subject),FUN = mean)
There were 50 or more warnings (use warnings() to see the first 50)
> head(data_Ag)
  Group.1 Subject Gender Group Age Medication Measure Glucose
1       1       1     NA    NA  65         NA     2.5  177.00
2       2       2     NA    NA  53         NA     2.5  175.25
3       3       3     NA    NA  60         NA     2.5  174.00
4       4       4     NA    NA  60         NA     2.5  171.50
5       5       5     NA    NA  55         NA     2.5  172.25
6       6       6     NA    NA  70         NA     2.5  171.00
```

aggregate

```
> data_Ag2 <- aggregate(data_D,by = list(data_D$Subject),FUN = function(x){
+   if(is.numeric(x)) return(mean(x))
+   if(is.factor(x)) return(unique(x))}
+ )
```

```
> head(data_Ag2)
  Group.1 Subject Gender    Group Age Medication Measure Glucose
1       1       1      F Diabetes  65   acarbose     2.5  177.00
2       2       2      M Diabetes  53   miglitol     2.5  175.25
3       3       3      M Diabetes  60   acarbose     2.5  174.00
4       4       4      M Diabetes  60   acarbose     2.5  171.50
5       5       5      F Diabetes  55   miglitol     2.5  172.25
6       6       6      M Diabetes  70   miglitol     2.5  171.00
```

with

- ***with()*** and ***within()*** <- Evaluate an R expression in an environment constructed from data, possibly modifying (a copy of) the original data.

```
data2 <- within(data,expr = IMC<- Mother_Weight/(Mother_Height/100)^2)
```

```
> str(data2)
```

with

- ***with()*** and ***within()*** <- Evaluate an R expression in an environment constructed from data, possibly modifying (a copy of) the original data.

```
data2 <- within(data,expr = IMC<- Mother_Weight/(Mother_Height/100)^2)
```

```
> str(data2)
'data.frame': 250 obs. of  11 variables:
 $ SubjectID          : Factor w/ 250 levels "Subject001","Subject002",..: 1 2 3 4 5 6 7 8 9 10 ...
 $ Mother_Age         : int  31 25 33 37 27 28 31 32 19 28 ...
 $ Mother_Weight      : int  55 66 58 53 56 58 59 68 84 58 ...
 $ Mother_Height      : int  165 160 172 172 162 158 157 162 159 171 ...
 $ Newborn_Gender     : int  0 1 0 1 1 0 0 1 1 0 ...
 $ Delivery_Methods   : Factor w/ 4 levels "","Cesarian",..: 3 4 3 4 4 3 4 4 4 4 ...
 $ Beginning_of_labour: Factor w/ 3 levels "","Induced","Spontaneous": 2 2 3 3 2 2 3 3 2 3 ...
 $ Gestational_Age    : int  39 41 40 38 40 40 40 36 40 39 ...
 $ Newborn_Weight     : int  2805 3525 3830 3395 3290 3175 3215 965 2835 2860 ...
 $ Newborn_Height     : int  48 52 50 51 51 50 48 33 51 48 ...
 $ IMC                : num  0.00202 0.00258 0.00196 0.00179 0.00213 ...
```

Graphics

Graphical Devices

Device	Description
windows	the graphics device for Windows (on screen)
quartz	the graphics device for the macOS native Quartz 2d graphics system
pdf	Write PDF graphics commands to a file
bmp	BMP bitmap device
png	PNG bitmap device
jpeg	JPEG bitmap device
tiif	TIFF bitmap device

- ***windows(*****width, height, pointsize, record, rescale, xpinch, ypinch, bg, canvas, gamma, xpos, ypos, buffered, title, restoreConsole, clickToConfirm, fillOddEven, family, antialias*****)***
- ***win.metafile(*****filename = "", width = 7, height = 7, pointsize = 12, family, restoreConsole = TRUE*****)***
- ***savePlot(*****filename = paste0("Rplot.", type), type = c("png", "jpeg", "tiff", "bmp"), device = dev.cur()*****)***

Graphical Devices

Device	Description
windows	the graphics device for Windows (on screen)
quartz	the graphics device for the macOS native Quartz 2d graphics system
pdf	Write PDF graphics commands to a file
bmp	BMP bitmap device
png	PNG bitmap device
jpeg	JPEG bitmap device
tiif	TIFF bitmap device

- ***quartz(*****title, width, height, pointsize, family, antialias, type, file = NULL, bg, canvas, dpi*****)***
- ***quartz.save(*****file, type = "png", device = dev.cur(), dpi = 100, ...*****)***
- ***##### Close all graphs***
- ***graphics.off()***

Graphical Devices

- ***pdf(****file = if(onefile) "Rplots.pdf" else "Rplot%03d.pdf", width, height, onefile, family, title, fonts, bg, fg, pointsize, pagecentre,…****)***
- ***bmp(****filename = "Rplot%03d.bmp", width = 480, height = 480, units = "px", pointsize = 12, bg = "white", res = NA, family = "", restoreConsole = TRUE,…****)***
- ***jpeg(****filename = "Rplot%03d.jpeg", width = 480, height = 480, units = "px", pointsize = 12, quality = 75, bg = "white", res = NA, family = "",…****)***
- ***png(****filename = "Rplot%03d.png", width = 480, height = 480, units = "px", pointsize = 12, bg = "white", res = NA, family = "",…****)***
- ***tiff(****filename = "Rplot%03d.tiff", width = 480, height = 480, units = "px", pointsize = 12, compression = c("none", "rle", "lzw", "jpeg", "zip", "lzw+p", "zip+p"), bg = "white", res = NA, family = "",…****)***
- ***#####close***
- ***dev.off()***

plot

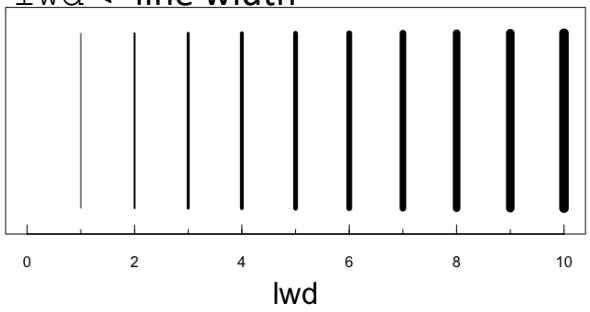
- ***plot(x, y, …)***

- `main` <- an overall title for the plot
- `sub` <- a sub title for the plot
- `xlab` <- a lable for the x axis
- `ylab` <- a lable for the y axis
- `xlim; ylim` <- axes limits
- `axes (xaxt, yaxt)` <- logical value indicating whether both (or each) axes should be drawn

plot

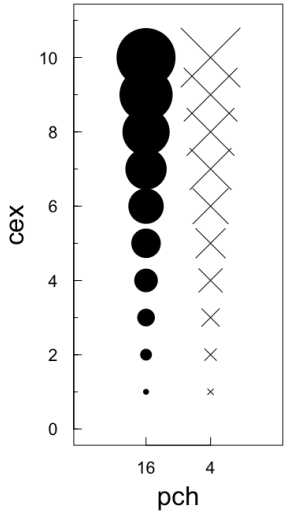
- `lty` <- line type

- `lwd` <- line width



- `pch` <- controls the shape of points

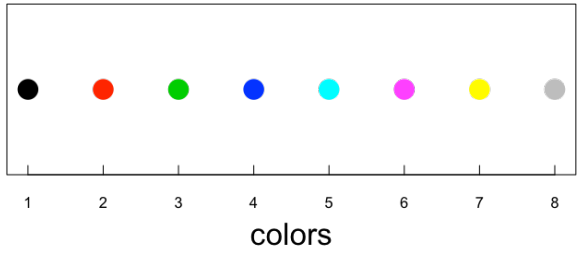
- `cex` ("character expansion") <- size of points.



Color

- `col` <- color of the points/line.
- `col.axis` <- color for axis annotation
- `col.lab` <- color for x and y labels
- `fg` <- plot foreground color (axes, boxes - also sets col= to same)
- `bg` <- background (fill) color for the open plot symbols

- integer



- Name

http://www.stat.columbia.edu/~tzheng/files/Rcolor.pdf

```
> colors()
```

- RGB
  - Eg. Black = “#000000”

fonts

- `ps` <- The font pointsize
- `cex` <- The "character expansion" (text size = ps*cex)
- `font` <- The font "face" (1=plain, 2=bold, 3=italic, 4=bold-italic)

- `family`

```
> windowsFonts()
```

```
> quartzFonts()
```

fonts

- ***windowsFont(family)***
- ***windowsFonts(...)***
- handle the translation of a device-independent R graphics font family name to a windows font description and are available only on Windows.

```
> windowsFonts()

$serif

[1] "Times-Roman"      "Times-Bold"
"Times-Italic"     "Times-BoldItalic"

$sans

[1] "Helvetica"             "Helvetica-
Bold"        "Helvetica-Oblique"
"Helvetica-BoldOblique"

$mono

[1] "Courier"               "Courier-Bold"
"Courier-Oblique"     "Courier-
BoldOblique"

> windowsFonts(Arial = c("Arial","Arial-
Italic","Arial-Bold","Arial-
BoldItalic"))
```

fonts

- ***quartzFont(family)***
- ***quartzFonts()***

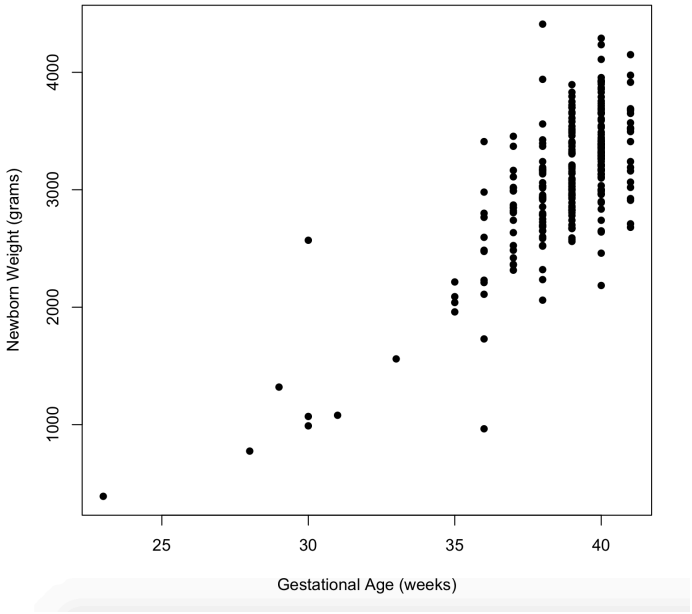
- handle the translation of a device-independent **R** graphics font family name to a quartz font description.
- They are only available on Unix-alikes, i.e, not on Windows, and typically used on the Mac.

```
> quartzFonts()

$serif

[1] "Times-Roman"      "Times-Bold"
"Times-Italic"     "Times-BoldItalic"

$sans

[1] "Helvetica"             "Helvetica-
Bold"        "Helvetica-Oblique"
"Helvetica-BoldOblique"

$mono

[1] "Courier"               "Courier-Bold"
"Courier-Oblique"     "Courier-
BoldOblique"

> quartzFonts(Arial = c("Arial","Arial-
Italic","Arial-Bold","Arial-
BoldItalic"))
```

Plots

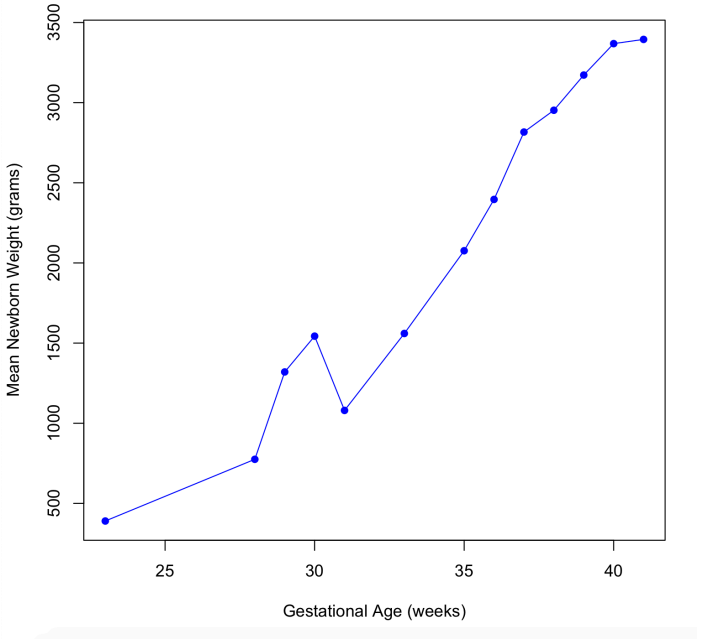
```
> data_Obst <-
read.csv("Dataset_Obst_Example1.csv")
```

```
> plot(data_Obst$Gestational_Age,
data_Obst$Newborn_Weight, pch=16,
xlab="Gestational Age (weeks)",
ylab="Newborn Weight (grams)")
```



Plots

```
> m <-
aggregate(data$Newborn_Weight,by
=
list(data$Gestational_Age),FUN=me
an)

> plot(as.integer(names(m)),m,

type="o",pch=16,col="blue",

xlab="Gestational Age (weeks)",

ylab="Mean Newborn Weight
(grams)")
```



Formula

- `formula <- of the form y ~ x or to plot by groups: y ~ x | z,` `where z evaluates to a factor or other variable dividing the data into groups.`

- Symbols:
- “+” inclusion
- “-” exclusion
- “.” include all variables
- “*” combination of factors

Formula

- `formula <- of the form y ~ x or to plot by groups: y ~ x | z, where z evaluates to a factor or other variable dividing the data into groups.`

- Symbols:
- “+” inclusion
- “-” exclusion
- “.” include all variables
- “*” combination of factors