Basic R

CITS4009 Computational Data Analysis

Dr. Mubashar Hassan

Department of Computer Science and Software Engineering
The University of Western Australia

Semester 2, 2024

Subsetting

Individual elements of a vector, matrix, array or data frame are accessed with “`[ ]`” by specifying their index, or their name

```
a <- data.frame(row.names = "mpg" %+% chr(1:nrow(mpg)),
                manufacturer = mpg$manufacturer,
                model = mpg$model, displ = mpg$displ,
                cyl = mpg$cyl)
head(a)
```

```
##      manufacturer model displ cyl
## mpg1         audi    a4   1.8   4
## mpg2         audi    a4   1.8   4
## mpg3         audi    a4   2.0   4
## mpg4         audi    a4   2.0   4
## mpg5         audi    a4   2.8   6
## mpg6         audi    a4   2.8   6
```

Subsetting

- By index, by row names and column names

```
a[3,3]
## [1] 2
```

```
a["mpg3", "displ"]
## [1] 2
```

```
a["mpg3",]
```

```
##      manufacturer model displ cyl
## mpg3         audi    a4     2   4
```

Subsetting

- Subset rows by a vector of indices

```
a[c(1:2),]
```

```
##      manufacturer model displ cyl
## mpg1         audi    a4   1.8   4
## mpg2         audi    a4   1.8   4
```

```
a[-c(2:nrow(mpg)), ]
```

the - exclude the row from 2 to rest of the row

```
##      manufacturer model displ cyl
## mpg1         audi    a4   1.8   4
```

Subsetting

- Subset rows by a logical vector

```
a[c(T,F,T),]
```

```
##         manufacturer                       model displ cyl
## mpg1            audi                          a4   1.8   4
## mpg3            audi                          a4   2.0   4
## mpg4            audi                          a4   2.0   4
## mpg6            audi                          a4   2.8   6
## mpg7            audi                          a4   3.1   6
## mpg9            audi                  a4 quattro   1.8   4
## mpg10           audi                  a4 quattro   2.0   4
## mpg12           audi                  a4 quattro   2.8   6
## mpg13           audi                  a4 quattro   2.8   6
## mpg15           audi                  a4 quattro   3.1   6
## mpg16           audi                  a6 quattro   2.8   6
## mpg18           audi                  a6 quattro   4.2   8
```

Subsetting

- Subset columns

```
a$manufacturer
```

```
##   [1] "audi"       "audi"       "audi"       "audi"       '
##   [9] "audi"       "audi"       "audi"       "audi"       '
##  [17] "audi"       "audi"       "chevrolet"  "chevrolet"  '
##  [25] "chevrolet"  "chevrolet"  "chevrolet"  "chevrolet"  '
##  [33] "chevrolet"  "chevrolet"  "chevrolet"  "chevrolet"  '
##  [41] "dodge"      "dodge"      "dodge"      "dodge"      '
##  [49] "dodge"      "dodge"      "dodge"      "dodge"      '
##  [57] "dodge"      "dodge"      "dodge"      "dodge"      '
##  [65] "dodge"      "dodge"      "dodge"      "dodge"      '
##  [73] "dodge"      "dodge"      "ford"       "ford"       '
##  [81] "ford"       "ford"       "ford"       "ford"       '
##  [89] "ford"       "ford"       "ford"       "ford"       '
##  [97] "ford"       "ford"       "ford"       "honda"      '
```

Subsetting

- Comparison resulting in a logical vector

```
a$manufacturer == "audi"
```

remember

```
##   [1]  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE  TRUE
##  [20] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [39] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [58] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [77] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
##  [96] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [115] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [134] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [153] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [172] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [191] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [210] FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE FALSE
## [229] FALSE FALSE FALSE FALSE FALSE FALSE
```

Subsetting

- Subset the selected rows

```
a[a$manufacturer == "audi" & a$model == "a4 quattro",]
```

```
##        manufacturer      model displ cyl
## mpg8           audi a4 quattro   1.8   4
## mpg9           audi a4 quattro   1.8   4
## mpg10          audi a4 quattro   2.0   4
## mpg11          audi a4 quattro   2.0   4
## mpg12          audi a4 quattro   2.8   6
## mpg13          audi a4 quattro   2.8   6
## mpg14          audi a4 quattro   3.1   6
## mpg15          audi a4 quattro   3.1   6
```

Functions

Functions take data as *input*, process it into *output*

- **Input**: function arguments (0, 1, 2, ...)
- **Output**: function result (exactly one)

```
add <- function(a,b) {
    result <- a+b
  return(result)
}
```

Operators

Operators: Short-cut writing for frequently used functions of one or two arguments.

- Assignment

`<-`	assignment operator

- Arithmetic

`+`	addition	`-`	subtraction
`*`	multiplication	`/`	division
`^`	exponent	`%%`	mod
`%/%`	integer division	`%*%`	dot product or matrix multiplication

Operators (Why <-? Why not =?)

```r
x <- rnorm(100)
y <- 2*x + rnorm(100)
lm(formula=y~x)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##      0.2968       1.8606
```

- `<-` in the first two lines is used as an assignment operator;
- `=` in the third line does not serve as an assignment operator; instead, it is an operator that specifies a named parameter formula for the `lm` function.

Operators

- Set

`%in%`	set membership

- Logical

`&`	and
`|`	or
`!`	not

- Comparison

`<`	less than	`>`	greater than
`<=`	less or equal to	`>=`	greater or equal to
`==`	is equal to	`!=`	not equal to

Operators (cont.)

Sets can be represented as vectors in R. Example: check if each element in a set is a member of another set:

```
A <- letters[1:3]  # a vector containing ("a", "b", "c")
B <- letters[1:5]  # a vector containing ("a", "b", "c", "d", "e")
C <- letters[2:6]  # a vector containing ("b", "c", "d", "e", "f")
print(A %in% B)
## [1] TRUE TRUE TRUE
```

```
print(A %in% C)
## [1] FALSE  TRUE  TRUE
```

Can combine with the built-in `all` function to check if a set is a subset of another set:

```
print(all(A %in% B))   # is A a subset of B?
## [1] TRUE
```

```
print(all(A %in% C))   # is A a subset of C?
## [1] FALSE
```

Frequently used functions

- Basic stats (`max`, `min`, `summary`)
- Rounding (`round`, `floor`)
- Concatenate vectors (`c`, `cbind`, `rbind`)
- Size (`length`, `dim`, `nrow`, `ncol`)
- Vector sorting (`sort`, `rank`, `order`)
- Display or concatenate into a string (`print`, `cat`, `paste`, `format`)
- Others (`apply`, `table`, `which`)

```
LETTERS
##  [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K" "L" "M" "I
```

```
which( LETTERS == "R" )
## [1] 18
```

(Note that `LETTERS` and `letters` are built-in variables)

Examples

```r
a <- letters[1:5]
b <- table(a, sample(a))
b
```

```
##
## a   a b c d e
##   a 0 0 0 1 0
##   b 0 0 0 0 1
##   c 1 0 0 0 0
##   d 0 1 0 0 0
##   e 0 0 1 0 0
```

```r
apply(b, 1, mean)
```

```
##   a   b   c   d   e
## 0.2 0.2 0.2 0.2 0.2
```

Branching

```
if (logical_expression) {
  statements
} else {
  alternative_statements
}
```

The `else` part is optional. The braces `{ }` is optional if only one statement for the logical expression.

Branching Example

```r
x <- -4
if (x >= 0) {
  print(sqrt(x))
} else {
  print(NA)
}
```

```
## [1] NA
```

More Branching

```
ifelse (logical_expression, yes_statement, no_statement)
```

```r
x <- c(4:-4)
sqrt(ifelse(x >= 0, x, NA))
```

```
## [1] 2.000000 1.732051 1.414214 1.000000 0.000000       NA
```

Looping

When the same or similar tasks need to be performed multiple times; for all elements of a list; for all columns of an array; etc.

For loop:

```
for (i in 1:5) {
   print(i*i)
}
## [1] 1
## [1] 4
## [1] 9
## [1] 16
## [1] 25
```

Looping

`While` loop:

```
i <- 1
while (i <= 5) {
   print(i*i)
   i <- i + sqrt(i)
}
## [1] 1
## [1] 4
## [1] 11.65685
```

Can also use `repeat` for looping also (need to combine with `break` to terminate looping):

```
i <- 1
repeat {
  if (i > 5) break
  print(i*i)
  i <- i + sqrt(i)
}
## [1] 1
## [1] 4
## [1] 11.65685
```

Looping

When R sees the `break` statement, it will pass control to the statement immediately after the loop. In contrary, when it sees the `next` statement, it will pass control to the beginning of the next round of the loop.

```
# Print the square of each odd number in 1..10
for (i in 1:10) {
  if (i %% 2 == 0)
      next
  print(i*i)
}
## [1] 1
## [1] 9
## [1] 25
## [1] 49
## [1] 81
```

References

- **Practical Data Science with R**, *Nina Zumel, John Mount*, Manning, 2nd Ed., 2020
- **R for Data Science (2e)**, *Hadley Wickham, Garrett Grolemund*, O'Reilly, 2023 (Chapter 2)
- **Introduction to the R language:** https://users.soe.ucsc.edu/~lshiue/bioc/Rintro.ppt
- **An Introduction to R:** http://csg.sph.umich.edu/abecasis/class/815.04.pdf
- **Differences between assignment operators in R:** https://renkun.me/2014/01/28/difference-between-assignment-operators-in-r/

Data at a Glance

CITS4009 Computational Data Analysis

Dr. Mubashar Hassan

Department of Computer Science and Software Engineering
The University of Western Australia

Semester 2, 2024

The Customer Dataset

Synthetic example data derived from Census PUMS data to predict the probability of health insurance coverage.

Data can be obtained from:
https://github.com/WinVector/zmPDSwR/tree/master/Custdata

```
custdata <- read.table('custdata.tsv', header=T, sep='\t')
```

Customer Data Structure

```r
str(custdata)
```

```
## 'data.frame':    1000 obs. of  11 variables:
##  $ custid      : int  2068 2073 2848 5641 6369 8322 852
##  $ sex         : chr  "F" "F" "M" "M" ...
##  $ is.employed : logi  NA NA TRUE TRUE TRUE TRUE ...
##  $ income      : int  11300 0 4500 20000 12000 180000 12
##  $ marital.stat: chr  "Married" "Married" "Never Marrie
##  $ health.ins  : logi  TRUE TRUE FALSE FALSE TRUE TRUE
##  $ housing.type: chr  "Homeowner free and clear" "Rente
##  $ recent.move : logi  FALSE TRUE TRUE FALSE TRUE FALSE
##  $ num.vehicles: int  2 3 3 0 1 1 1 3 2 1 ...
##  $ age         : num  49 40 22 22 31 40 39 48 44 70 ...
##  $ state.of.res: chr  "Michigan" "Florida" "Georgia" "Ne
```

Customer Data Summary

```r
summary(custdata)
```

```
##      custid              sex              is.employed
##  Min.   :   2068   Length:1000        Mode :logical   Mi
##  1st Qu.: 345667   Class :character   FALSE:73        1s
##  Median : 693403   Mode  :character   TRUE :599       Me
##  Mean   : 698500                      NA's :328       Me
##  3rd Qu.:1044606                                      3r
##  Max.   :1414286                                      Ma
##
##  housing.type        recent.move      num.vehicles
##  Length:1000        Mode :logical   Min.   :0.000   Min.
##  Class :character   FALSE:820       1st Qu.:1.000   1st
##  Mode  :character   TRUE :124       Median :2.000   Medi
##                     NA's :56        Mean   :1.916   Mear
##                                     3rd Qu.:2.000   3rd
##                                     Max.   :6.000   Max.
##                                     NA's   :56
```

Using Summary Statistics to spot problems

In R, you'll typically use the `summary()` command to take your first look at the data.

The goal is to understand whether you have the kind of customer information that

- can potentially help you predict health insurance coverage, and
- whether the data is of good enough quality to be informative.

Looking for several common issues:

- Missing values
- Invalid values and outliers
- Data ranges that are too wide or too narrow
- The units of the data

Read the summary

```
is.employed          income
Mode :logical    Min.    : -8700
FALSE:73         1st Qu.: 14600
TRUE :599        Median : 35000
NA's :328        Mean   : 53505
                 3rd Qu.: 67000
                 Max.   :615000

marital.stat
Divorced/Separated:155
Married           :516
Never Married     :233
Widowed           : 96

health.ins
Mode :logical
FALSE:159
TRUE :841
NA's :0

housing.type
Homeowner free and clear    :157
Homeowner with mortgage/loan:412
Occupied with no rent       : 11
Rented                      :364
NA's                        : 56

recent.move       num.vehicles
Mode :logical    Min.   :0.000
FALSE:820        1st Qu.:1.000
TRUE :124        Median :2.000
NA's :56         Mean   :1.916
                 3rd Qu.:2.000
                 Max.   :6.000
                 NA's   :56

age               state.of.res
Min.   :  0.0    California  :100
1st Qu.: 38.0    New York    : 71
Median : 50.0    Pennsylvania: 70
Mean   : 51.7    Texas       : 56
3rd Qu.: 64.0    Michigan    : 52
Max.   :146.7    Ohio        : 51
                 (Other)     :600
```

The variable is.employed is missing for about a third of the data. The variable income has negative values, which are potentially invalid.

About 84% of the customers have health insurance.

The variables housing.type, recent.move, and num.vehicles are each missing 56 values.

The average value of the variable age seems plausible, but the minimum and maximum values seem unlikely. The variable state.of.res is a categorical variable; summary() reports how many customers are in each state (for the first few states).

Exploratory Data Analysis

CITS4009 Computational Data Analysis

Dr. Mubashar Hassan

Department of Computer Science and Software Engineering
The University of Western Australia

Semester 2, 2024

Section 1

Visualisation

What's visualisation

Pictures are often better than text.

> *We cannot expect a small number of numerical values [summary statistics] to consistently convey the wealth of information that exists in data. Numerical reduction methods do not retain the information in the data. -William Cleveland*

The use of graphics to examine data is called *visualization*.

William Cleveland's Graphic Philosophe

- A fine balancing act
  - A graphic should display as much information as it can, with the lowest possible cognitive strain to the viewer.
- Strive for clarity. Make the data stand out. Specific tips for increasing clarity include
  - Avoid too many superimposed elements, such as too many curves in the same graphing space.
  - Find the right aspect ratio and scaling to properly bring out the details of the data.
  - Avoid having the data all skewed to one side or the other of your graph.
- Visualization is an iterative process. Its purpose is to answer questions about the data.
  - Different graphics are best suited for answering different questions.

Section 2

Exploratory Data Analysis (EDA)

What is exploratory data analysis?

Exploratory data analysis, or EDA for short is a task that use visualisation and transformation to explore your data in a systematic way.

EDA is an iterative cycle that involves

- Generate questions about your data.
- Search for answers by *visualising*, *transforming*, and *modelling* your data.
- Use what you learn to refine your questions and/or generate new questions.

EDA is not a formal process with a strict set of rules. More than anything, EDA is a state of mind.

The **goal** during EDA is to develop an understanding of your data.

The easiest way to achieve the goal is to use questions as tools to guide your investigation

How to ask good questions?

EDA is fundamentally a creative process.

Like most creative processes, the key to asking *quality* questions is to generate a large *quantity* of questions.

Two types of questions will always be useful for making discoveries within your data:

- What type of *variation* occurs within my variables?
- What type of *covariation* occurs between my variables?

Terms used in EDA

- A **variable** is a quantity, quality, or property that you can measure.
- A **value** is the state of a variable when you measure it. The value of a variable may change from measurement to measurement.
- An **observation** is a set of measurements made under similar conditions (you usually make all of the measurements in an observation at the same time and on the same object).
  - An observation will contain several values, each associated with a different variable.
  - An observation is also referred to as a data point.
- **Tabular data** is a set of values, each associated with a variable and an observation. Tabular data is tidy if each value is placed in its own "cell", each variable in its own column, and each observation in its own row.

What to look for in histograms and bar charts?

In both bar charts and histograms, tall bars show the common values of a variable, and shorter bars show less-common values.

Places that do not have bars reveal values that were not seen in your data.

To turn this information into useful questions, look for anything unexpected:

- Which values are the most common? Why?
- Which values are rare? Why? Does that match your expectations?
- Can you see any unusual patterns? What might explain them?

Does the data form subgroups?

Clusters of similar values suggest that subgroups exist in your data. To understand the subgroups, ask:

- How are the observations within each cluster similar to each other?
- How are the observations in separate clusters different from each other?
- How can you explain or describe the clusters?
- Why might the appearance of clusters be misleading?

Comparing two or more varialbes

- *Variation* describes the behavior within a variable,
- *Covariation* describes the behavior between variables.

Covariation is the tendency for the values of two or more variables to vary together in a related way.

The best way to spot covariation is to visualise the relationship between two or more variables. How you do that should again depend on the type of variables involved.

- a continuous variable and a categorical (categorical variable can be used as lengend, aesthetic mapping)
- two categorical variables (`geom_count` and `geom_tile`)
- two continuous variables (`geom_point` and `geom_boxplot` and `geom_bin2d` or `geom_hex`)

Questions to ask for Covariation

Patterns in your data provide clues about relationships. If a systematic relationship exists between two variables it will appear as a pattern in the data. If you spot a pattern, ask yourself:

- Could this pattern be due to coincidence (i.e. random chance)?
- How can you describe the relationship implied by the pattern?
- How strong is the relationship implied by the pattern?
- What other variables might affect the relationship?
- Does the relationship change if you look at individual subgroups of the data?

References

- **Practical Data Science with R**. By Nina Zumel and John Mount, Manning, 2014. (Chapter 3)
- **R for Data Science (2e)**. By Garret Grokemund and Hardley Wickham, O'Relly, 2023. (Chapter 3)
- Introduction to the R language: https://users.soe.ucsc.edu/~lshiue/bioc/Rintro.ppt
- An Introduction to R: http://csg.sph.umich.edu/abecasis/class/815.04.pdf
- Differences between assignment operators in R: https://renkun.me/2014/01/28/difference-between-assignment-operators-in-r/

Introduction to ggplot

CITS4009 Computational Data Analysis

Dr. Mubashar Hassan

Department of Computer Science and Software Engineering
The University of Western Australia

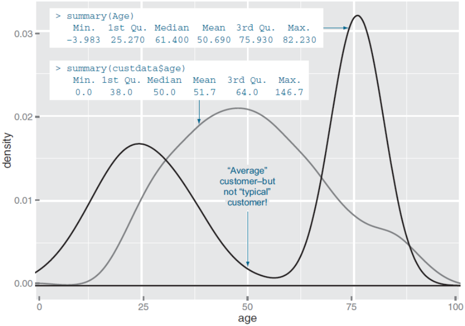
Semester 2, 2024

Section 1

Single Variable Plots

Distribution of a single variable

- What is the peak value of the distribution?
- How many peaks are there in the distribution (unimodality versus bimodality)?
- How normal (or lognormal) is the data?
- How much does the data vary? Is it concentrated in a certain interval or in a certain category?



Plots for single variable distribution

Graph Type	Uses
Histogram	Examines data range
Density Plot	Checks number of modes Checks if distribution is normal/lognormal/etc
Boxplot	Checks for anomalies and outliers
Bar Chart	Compares relative or absolute frequencies of the values of a categorical variable

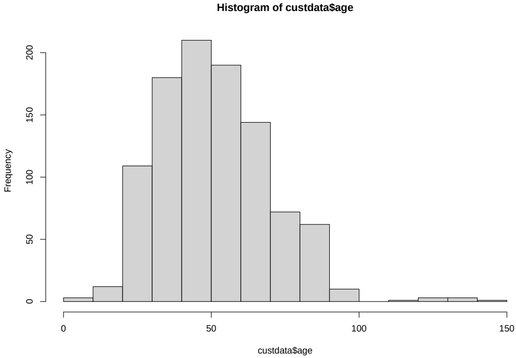
Section 2

Histograms

Histograms - the `hist` function in R

A basic histogram bins a variable into fixed-width buckets and returns the number of data points that falls into each bucket.

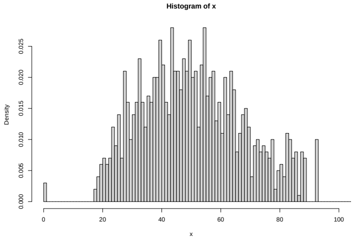
```
custdata <- read.table('custdata.tsv',header=T, sep='\t')
hist(custdata$age)
```



Histogram: Other useful options

- `breaks`: takes a sequence to specify where the breaks are
- `xlim`: takes the start and end point of x axis
- `freq`: `TRUE` for raw counts; `FALSE` for density (normalized by the total count), and the areas of the bars add to 1. This is called "density plot" in ggplot, except it is not a continuous line plot.

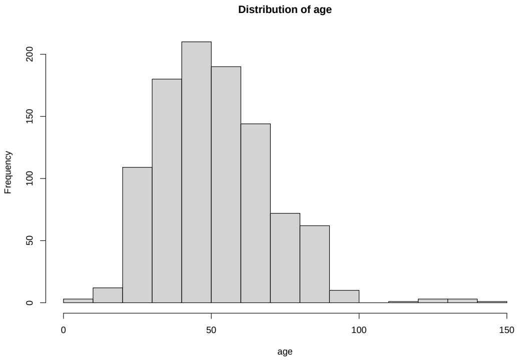
```
x <- custdata$age
hist(x, breaks=seq(0,150,1), xlim=c(0,100), freq = FALSE)
```



Adding titles

- Using Attributes of the function

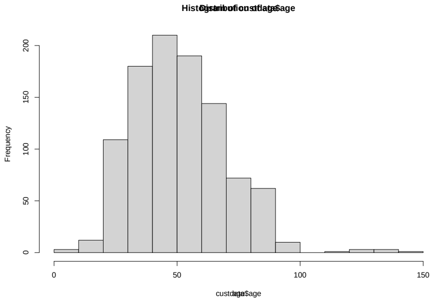
```
hist(custdata$age,main="Distribution of age",xlab="age")
```



Adding titles

- Using the `title()` function

```
hist(custdata$age)
title('Distribution of age',xlab='age')
```



To remove the default title from `hist`, do: `hist(custdata$age, main="")`

Section 3

A layered grammar of graphs - ggplot

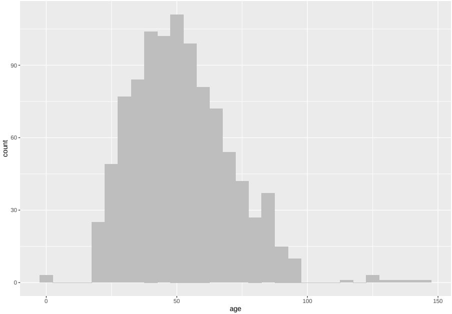
ggplot

- R has several systems for making graphs, but `ggplot2` is one of the most elegant and most versatile libraries.
- `ggplot2` implements the **grammar of graphics**, a coherent system for describing and building graphs.
- Begin a plot with the function `ggplot()`, which takes the data and create a coordinate system that you can add **layers** to.
- A reusable template for making graphs with `ggplot2` is given below. To make a graph, replace the bracketed parts in the code below with
  - a dataset,
  - a geom function (chart type), or
  - a collection of mappings (data selection for each coordinate).

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

Histograms (ggplot2)

```
library(ggplot2)
ggplot(data = custdata) +
  geom_histogram(mapping = aes(x=age),
                 binwidth=5, fill="gray")
```

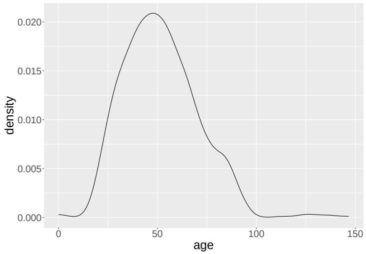


Density plot (ggplot2)

In ggplot, a **density plot** is a "continuous histogram" of a variable, except the area under the density plot is equal to 1.

- A point on a density plot corresponds to the fraction of data (or the percentage of data, divided by 100) that takes on a particular value.

```
library(ggplot2)
ggplot(custdata) + geom_density(aes(x=age)) +
    theme(text = element_text(size = 24))
```



References

- **Practical Data Science with R**. By Nina Zumel and John Mount, Manning, 2014. (Chapter 3)
- **R for Data Science (2e)**. By Garret Grokemund and Hardley Wickham, O'Relly, 2023. (Chapter 3)
- Introduction to the R language: https://users.soe.ucsc.edu/~lshiue/bioc/Rintro.ppt
- An Introduction to R: http://csg.sph.umich.edu/abecasis/class/815.04.pdf
- Differences between assignment operators in R: https://renkun.me/2014/01/28/difference-between-assignment-operators-in-r/