Data Cleaning, Data Visualization, & Functions

Fundamental Techniques in Data Science

Kyle M. Lang

Department of Methodology & Statistics
Utrecht University

Outline

FUNCTIONS

R Functions

Functions are the foundation of R programming.

- Other than data objects, almost everything else that you interact with when using R is a function.
- Any R command written as a word followed by parentheses `()` is a function.
  - `mean()`
  - `library()`
  - `mutate()`
- Infix operators are aliased functions.
  - `<-`
  - `+`, `-`, `*`
  - %>%, %$%, %<>%

User-Defined Functions

We can define our own functions using the `function()` function.

```r
square <- function(x) {
    out <- x^2
    out
}
```

After defining a function, we use it in the same way as any other R function.

```r
square(5)

[1] 25
```

User-Defined Functions

One-line functions don’t need braces

```
square <- function(x) x^2
square(5)

[1] 25
```

Function arguments are not strictly typed. R will try to work with whatever you provide as input.

```
square(1:5)

[1]  1  4  9 16 25

square(pi)

[1] 9.869604

square(TRUE)

[1] 1

square("bob") # But one can only try so hard

Error in x^2
```

User-Defined Functions

Functions can take multiple arguments.

```
mod <- function(x, y) x %% y
mod(10, 3)

[1] 1
```

Sometimes it’s useful to specify a list of arguments that we unpack inside the function.

```
getLsBeta <- function(datList) {
    X <- datList$X
    y <- datList$y

    solve(crossprod(X)) %*% t(X) %*% y
}

X       <- matrix(runif(500), ncol = 5)
datList <- list(y = X %*% rep(0.5, 5), X = X)

getLsBeta(datList = datList)

     [,1]
[1,]  0.5
[2
```

User-Defined Functions

Functions are first-class objects in R.

- We can treat them like any other R object.

R views an initialized, but unevaluated, function as a special object with type ”closure”

```
class(getLsBeta)

[1] "function"

typeof(getLsBeta)

[1] "closure"
```

After evaluation, functions are simply equivelent to the objects they return.

```
class(getLsBeta(datList))

[1] "matrix" "array"

typeof(getLsBeta(datList))

[1] "double"
```

User-Defined Functions

We can use functions as arguments to other operations and functions.

```r
fun1 <- function(x, y) x + y

## What will this command return?
fun1(1, fun1(1, 1))

[1] 3
```

Why would we care?

```r
s2 <- var(runif(100))
x  <- rnorm(100, 0, sqrt(s2))

X[1:10, ]

            [,1]       [,2]       [,3]      [,4]       [,5]
 [1,] 0.52431382 0.67136447 0.28228726 0.7148383 0.54204681
 [2,] 0.01926742 0.11693762 0.09148502 0.6929171 0.88371944
 [3,] 0.05100735 0.18432074 0.43547799 0.6097462 0.09026598
 [4,] 0.60566972 0.12944127 0.21000143 0.2441917 0.68141473
 [5,] 0.48737303 0.94030405 0.23988619 0.4915910 0.36353771
 [6,] 0.19941958 0.96670678 0.11455820 0.1243947 0.24253273
 [7,] 0.9550
```

DATA VISUALIZATION

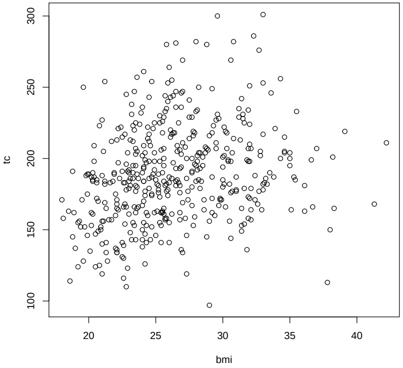
Setup

```r
dataDir  <- "../../../data/"

diabetes <- readRDS(paste0(dataDir, "diabetes.rds"))
titanic  <- readRDS(paste0(dataDir, "titanic.rds"))
bfi      <- readRDS(paste0(dataDir, "bfi.rds"))

## Convert surival indicator to a numeric dummy code:
titanic <- titanic %>% mutate(survived = as.numeric(survived) - 1)
```

Base R Graphics: Scatterplots

We can create a basic scatterplot using the `plot()` function.

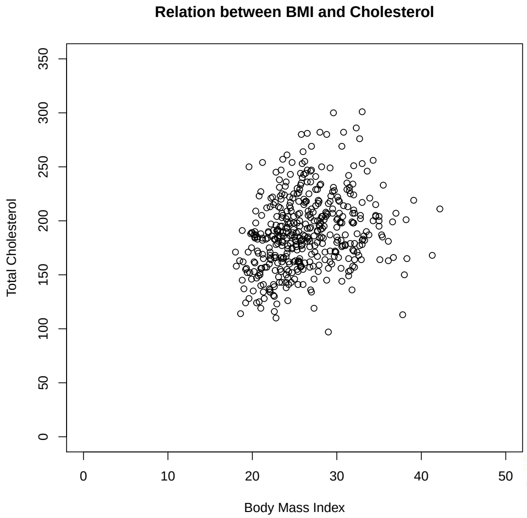
```
diabetes %$% plot(y = tc, x = bmi)
```



Base R Graphics: Scatterplots

```r
diabetes %$% plot(y = tc,
                  x = bmi,
                  ylab = "Total Cholesterol",
                  xlab = "Body Mass Index",
                  main = "Relation between BMI and Cholesterol",
                  ylim = c(0, 350),
                  xlim = c(0, 50)
                  )
```

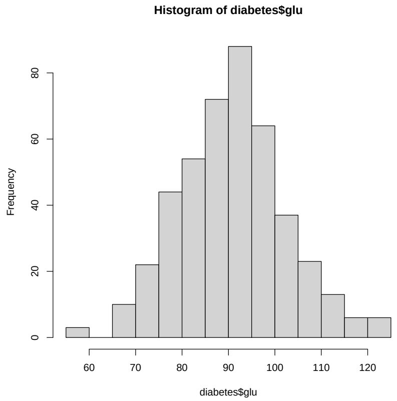
Base R Graphics: Scatterplots



Base R Graphics: Histograms

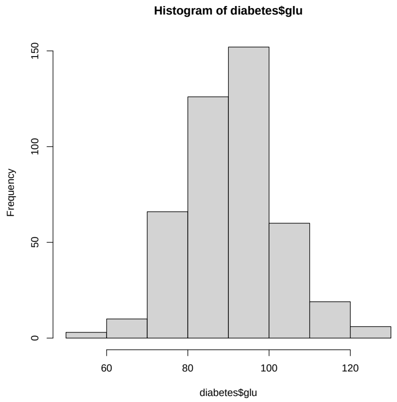
We can create a simple histogram with the `hist()` function.

```
hist(diabetes$glu)
```



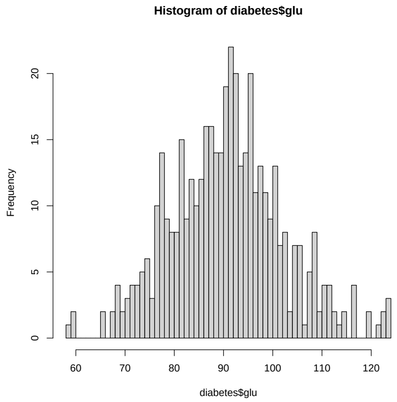
Base R Graphics: Histograms

```r
hist(diabetes$glu, breaks = 5)
```



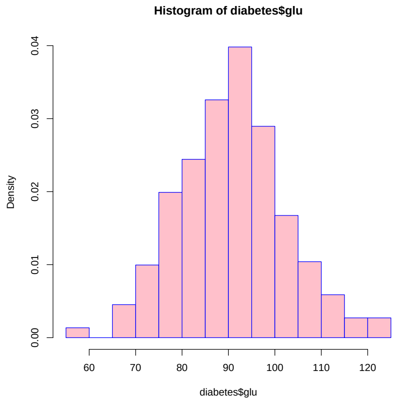
Base R Graphics: Histograms

```r
hist(diabetes$glu, breaks = 50)
```



Base R Graphics: Histograms

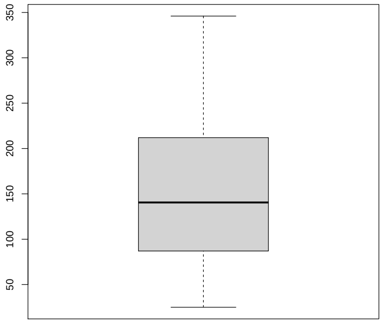
```
hist(diabetes$glu, col = "pink", border = "blue", probability = TRUE)
```



Base R Graphics: Boxplots

We can create simple boxplots via the `boxplot()` function.

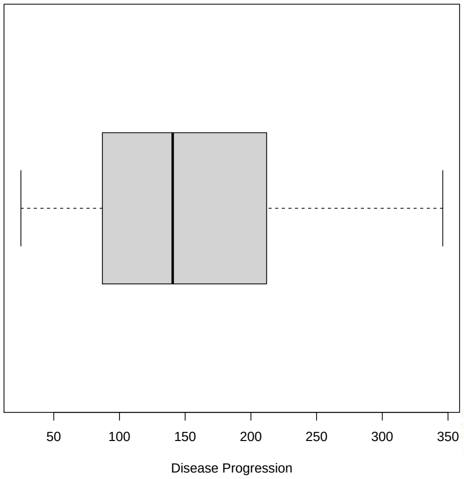
```r
boxplot(diabetes$progress)
```



Base R Graphics: Boxplots

```r
boxplot(diabetes$progress,
        horizontal = TRUE,
        range = 3,
        xlab = "Disease Progression")
```

Base R Graphics: Boxplots

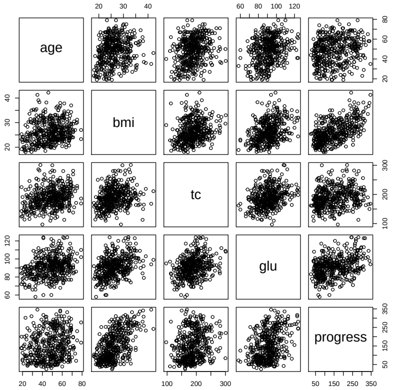


```
boxplot(prog
```

Base R Graphics: Fancy Things

Plotting an entire data frame produces a scatterplot matrix.

```
diabetes %>% select(age, bmi, tc, glu, progress) %>% plot()
```

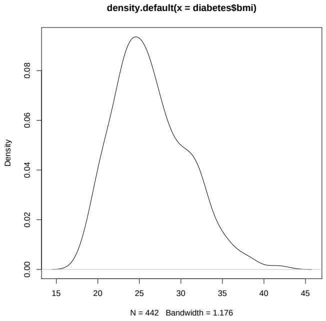


Base R Graphics: Fancy Things

The `density()` function estimates the density of a variable.

- If we plot a density object, we get a kernel density plot.

```
density(diabetes$bmi) %>% plot()
```

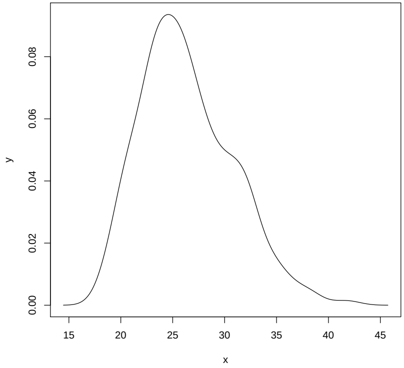


Base R Graphics: Fancy Things

```
d <- density(diabetes$bmi)
ls(d)

[1] "bw"          "call"        "data.name" "has.na"
[5] "n"           "x"           "y"
```

Base R Graphics: Fancy Things

```r
d %$% plot(y = y, x = x, type = "l")
```

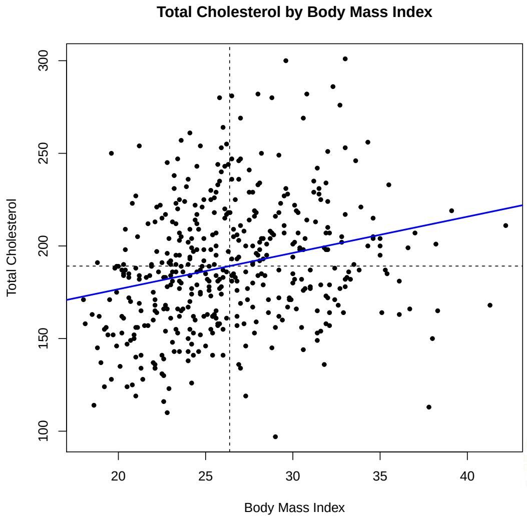


Base R Graphics: Workflow

Base R graphics work by building up graphics from layers.

```
## Start with a simple scatterplot:
diabetes %$% plot(y = tc, x = bmi, pch = 20, xlab = "", ylab = "")

## Use the abline() function to add lines representing the means of x and y:
abline(h = mean(diabetes$tc), v = mean(diabetes$bmi), lty = 2)

## Add the best fit line from a linear regression of 'tc'  onto 'bmi':
diabetes %$%
    lm(tc ~ bmi) %>%
    coef() %>%
    abline(coef = ., col = "blue", lwd = 2)

## Add titles:
title(main = "Total Cholesterol by Body Mass Index",
      ylab = "Total Cholesterol",
      xlab = "Body Mass Index")
```

Base R Graphics: Workflow

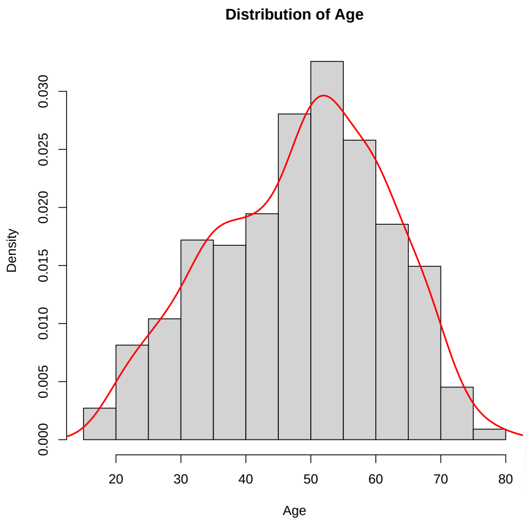


Base R Graphics: Workflow

Add a kernel density plot on top of a histogram.

```r
diabetes %$%
    hist(age,
         probability = TRUE,
         xlab = "Age",
         main = "Distribution of Age")

diabetes %$%
    density(age) %>%
    lines(col = "red", lwd = 2)
```

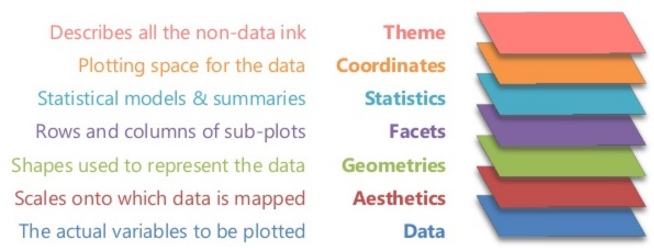
Base R Graphics: Workflow



GGPlot

Base R graphics are fine for quick-and-dirty visualizations (e.g., EDA, checking assumptions), but for publication quality graphics, we probably want to use GGPlot.

GGPlot uses the ”grammar of graphics” and ”tidy data” to build up a figure from modular components


Describes all the non-data ink
Theme
Plotting space for the data
Coordinates
Statistical models & summaries
Statistics
Rows and columns of sub-plots
Facets
Shapes used to represent the data
Geometries
Scales onto which data is mapped
Aesthetics
The actual variables to be plotted
Data

GGPlot: Basic Setup

We start by calling the `ggplot()` function.

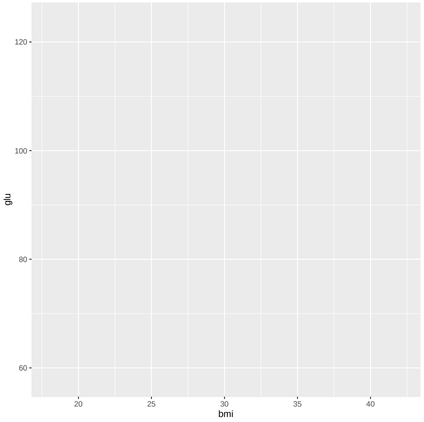
- We must define a data source.
- We must also give some aesthetic via the `aes()` function.

```r
library(ggplot2)
p1 <- ggplot(data = diabetes, mapping = aes(x = bmi, y = glu))
```

GGPlot: Basic Setup

At this point, our plot is pretty boring.

```
p1
```

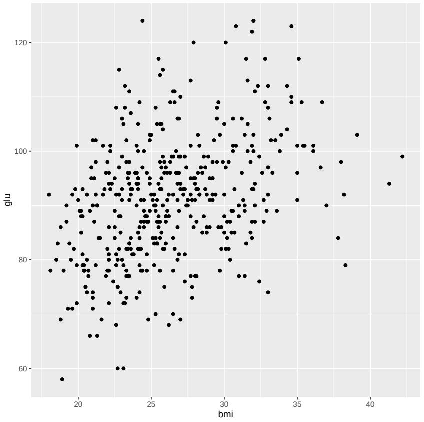


GGPlot: Geometries

We need to define some geometry via an appropriate `geom_X()` function.

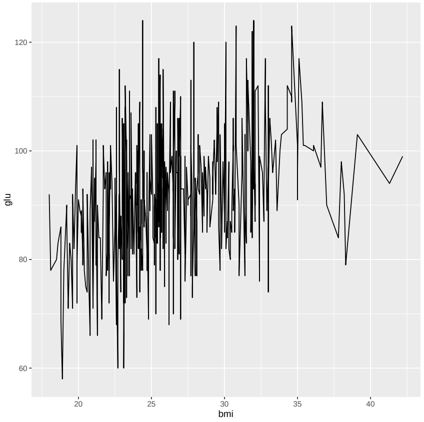
```
p1 + geom_point()
```

GGPlot: Geometries



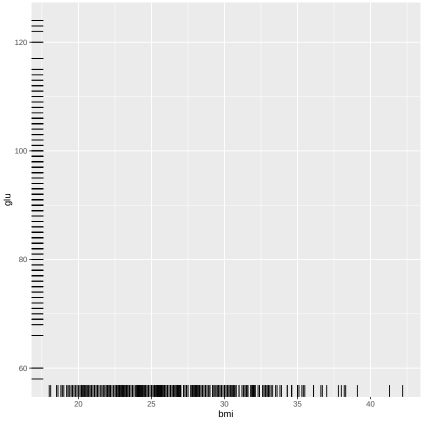
GGPlot: Geometries

```
p1 + geom_line()
```



GGPlot: Geometries

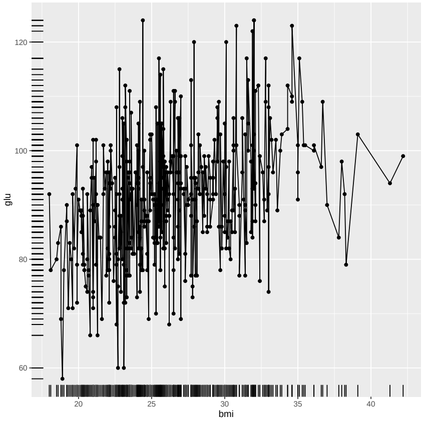
```r
p1 + geom_rug()
```



GGPlot: Geometries

We can also combine different geoms into a single figure

```r
p1 + geom_point() + geom_line() + geom_rug()
```

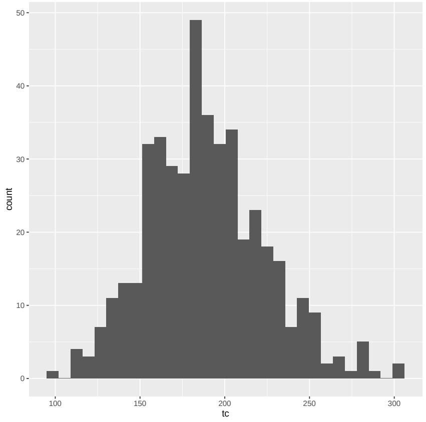


GGPlot: Geometries

We can use different flavors of geom for different types of data

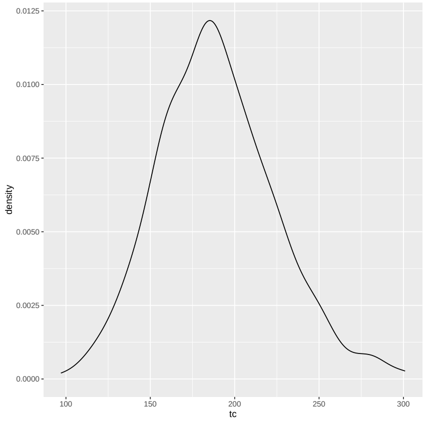
```r
p2 <- ggplot(diabetes, aes(tc))
p2 + geom_histogram()
```

GGPlot: Geometries



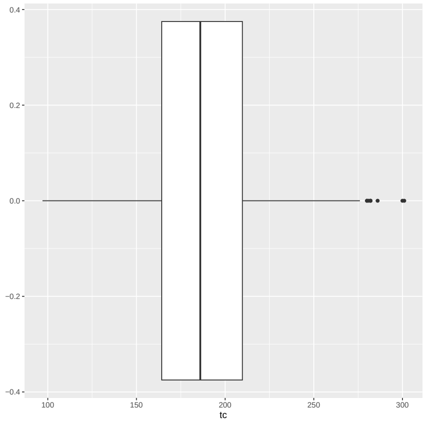
GGPlot: Geometries

```r
p2 + geom_density()
```



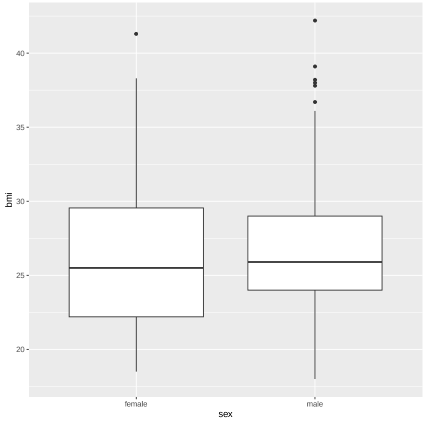
GGPlot: Geometries

```
p2 + geom_boxplot()
```



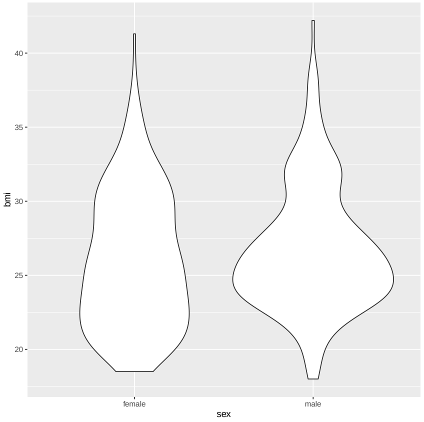
GGPlot: Geometries

```
p3 <- ggplot(diabetes, aes(sex, bmi))
p3 + geom_boxplot()
```



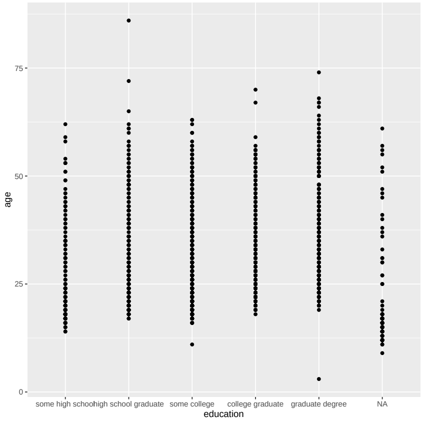
GGPlot: Geometries

```
p3 + geom_violin()
```



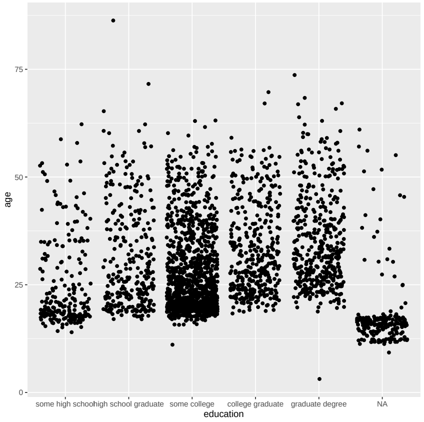
GGPlot: Geometries

```
p4 <- ggplot(bfi, aes(education, age))
p4 + geom_point()
```



GGPlot: Geometries

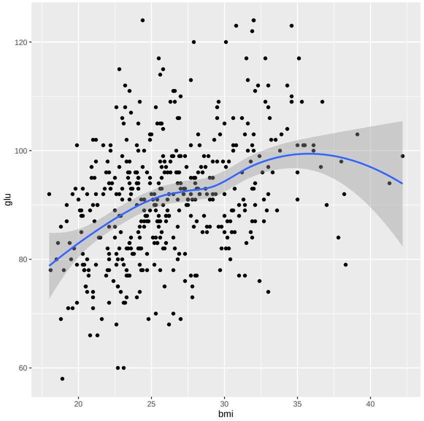
```
p4 + geom_jitter()
```



GGPlot: Statistics

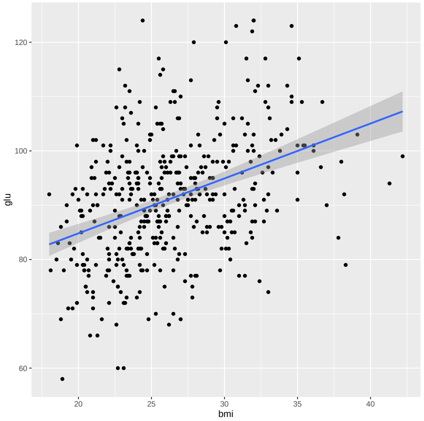
We can also add statistical summaries of the data

```
p1 + geom_point() + geom_smooth()
```



GGPlot: Statistics

```
p1 + geom_point() + geom_smooth(method = "lm")
```

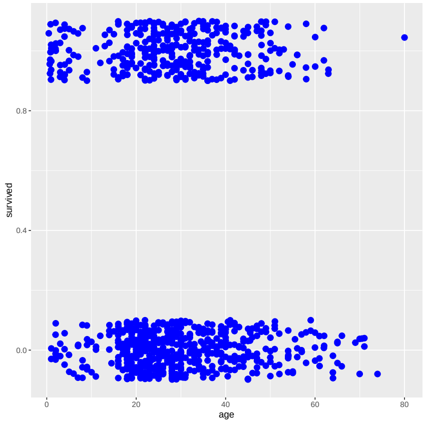


GGPlot: Styling

Changing style options outside of the `aes()` function applies the styling to the entire plot.

```r
p5 <- ggplot(titanic, aes(age, survived))
p5 + geom_jitter(color = "blue", size = 3, height = 0.1)
```

GGPlot: Styling

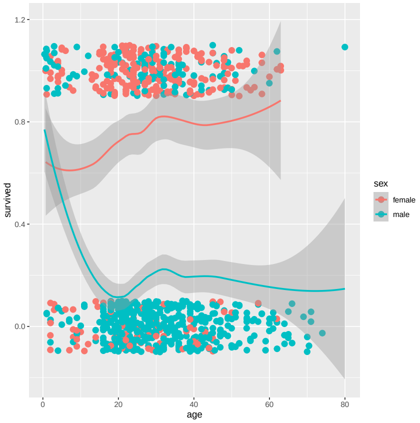


GGPlot: Styling

We can also apply styles as a function of variables by defining the style within the `aes()` function.

```
p6.1 <- ggplot(titanic, aes(age, survived, color = sex))
p6.1 + geom_jitter(size = 3, height = 0.1) + geom_smooth()
```

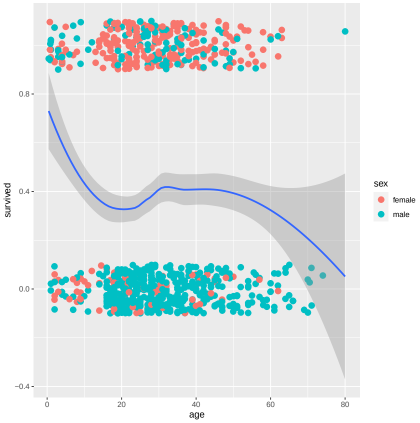
GGPlot: Styling



GGPlot: Styling

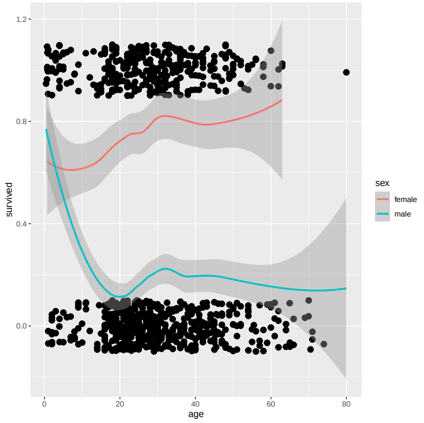
```
p6.2 <- ggplot(titanic, aes(age, survived))
p6.2 + geom_jitter(aes(color = sex), size = 3, height = 0.1) +
    geom_smooth()
```

GGPlot: Styling



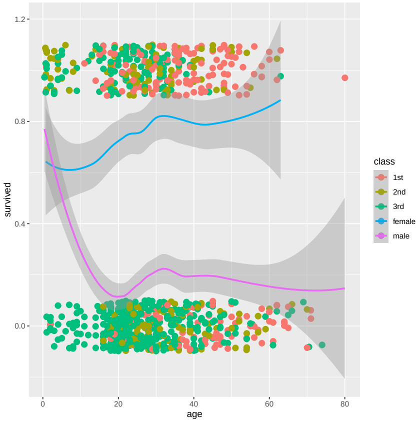
GGPlot: Styling

```
p6.2 + geom_jitter(size = 3, height = 0.1) +
    geom_smooth(aes(color = sex))
```



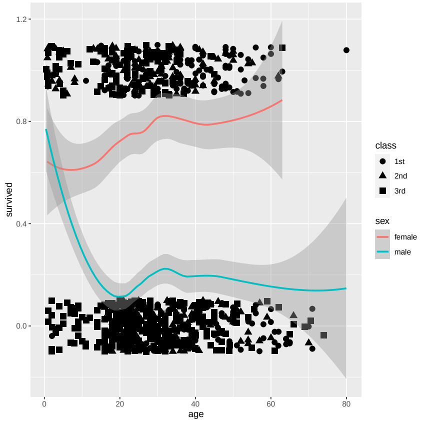
GGPlot: Styling

```
p6.2 + geom_jitter(aes(color = class), size = 3, height = 0.1) +
    geom_smooth(aes(color = sex))
```



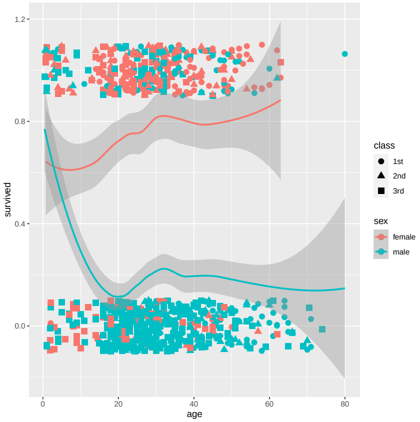
GGPlot: Styling

```
p6.2 + geom_jitter(aes(shape = class), size = 3, height = 0.1) +
  geom_smooth(aes(color = sex))
```



GGPlot: Styling

```
p6.1 + geom_jitter(aes(shape = class), size = 3, height = 0.1) +
  geom_smooth()
```

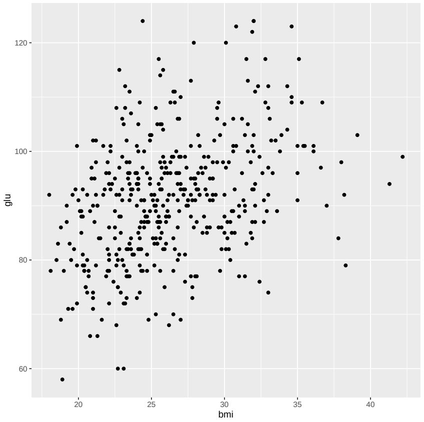


GGPlot: Themes

We can apply several pre-baked themes to adjust a plot's overall appearance

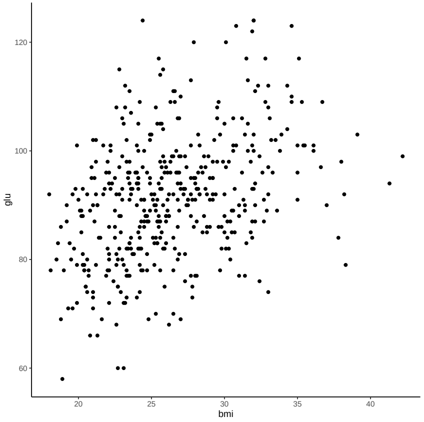
```
(p1.1 <- p1 + geom_point())
```

GGPlot: Themes



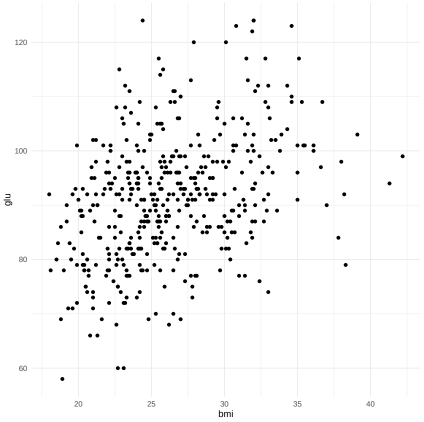
GGPlot: Themes

```
p1.1 + theme_classic()
```



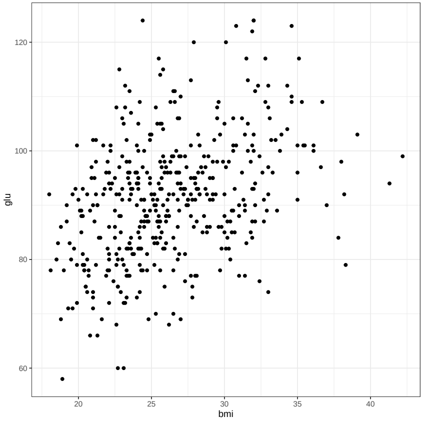
GGPlot: Themes

```
p1.1 + theme_minimal()
```



GGPlot: Themes

```
p1.1 + theme_bw()
```

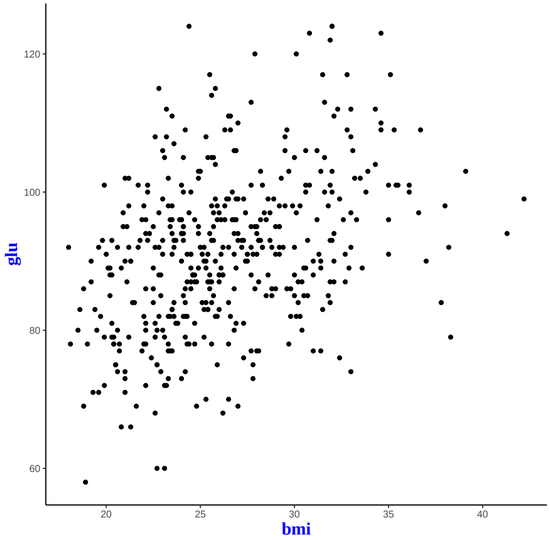


GGPlot: Themes

We can also moodifying individual theme elements.

```
p1.1 + theme_classic() +
    theme(axis.title = element_text(size = 16,
                                    family = "serif",
                                    face = "bold",
                                    color = "blue"),
          aspect.ratio = 1)
```

GGPlot: Themes

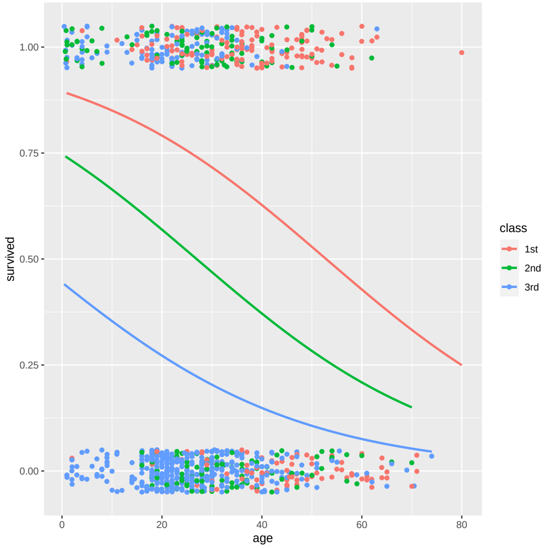


GGPlot: Facets

Facetting allow us to make arrays of conditional plots.

```
(p7 <- ggplot(titanic, aes(age, survived, color = class)) +
     geom_jitter(height = 0.05) +
     geom_smooth(method = "glm",
                 method.args = list(family = "binomial"),
                 se = FALSE)
)
```

GGPlot: Facets



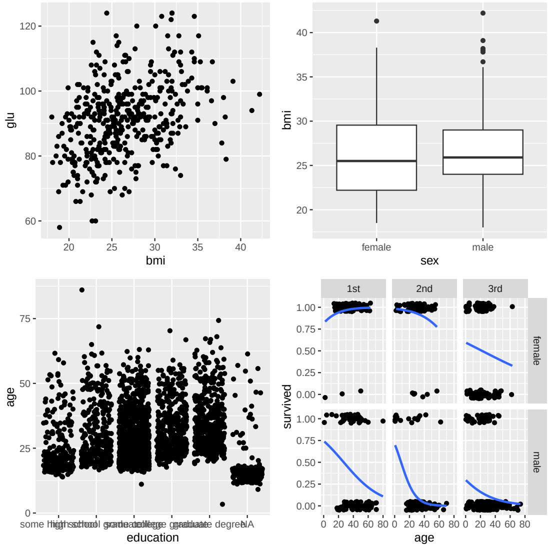
```
p7 + facet_w
```

GGPlot: Joining Multiple Figures

If we want to paste several different plots into a single figure (without facetting), we can use the utilities in the **gridExtra** package.

```
library(gridExtra)

grid.arrange(p1 + geom_point(),
             p3 + geom_boxplot(),
             p4 + geom_jitter(),
             p8 + facet_grid(vars(sex), vars(class)),
             ncol = 2)
```

GGPlot: Joining Multiple Figures



Saving Graphics

To save a graphic that we've created in R, we simply redirect the graphical output to a file using an appropriate function.

```
figDir <- "figures/"

## Save as PDF
pdf(paste0(figDir, "example_plot.pdf"))

p7 + facet_wrap(vars(sex))

dev.off()

pdf
  2
```

Saving Graphics

```r
## Save as JPEG
jpeg(paste0(figDir, "example_plot.jpg"))

p7 + facet_wrap(vars(sex))

dev.off()

pdf
  2
```

Saving Graphics

```
## Save as PNG
png(paste0(figDir, "example_plot.png"))

p7 + facet_wrap(vars(sex))

dev.off()

pdf
  2
```

Saving Graphics

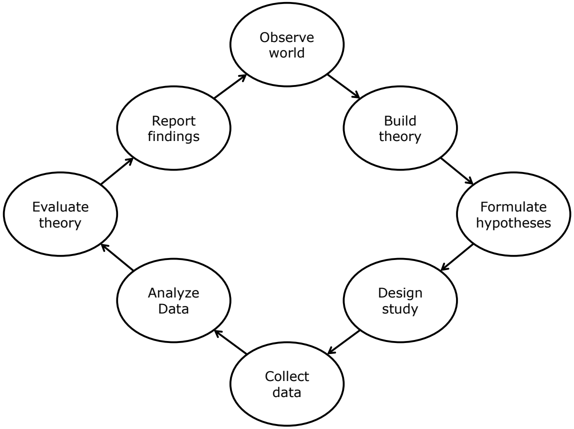
With PDF documents, we can save multiple figures to a single file.

```
pdf(paste0(figDir, "example_plot2.pdf"))

p6.1 + geom_jitter(size = 3, height = 0.1) + geom_smooth()
p7 + facet_wrap(vars(sex))
p8 + facet_grid(vars(sex), vars(class))

dev.off()

pdf
  2
```

Data Cleaning

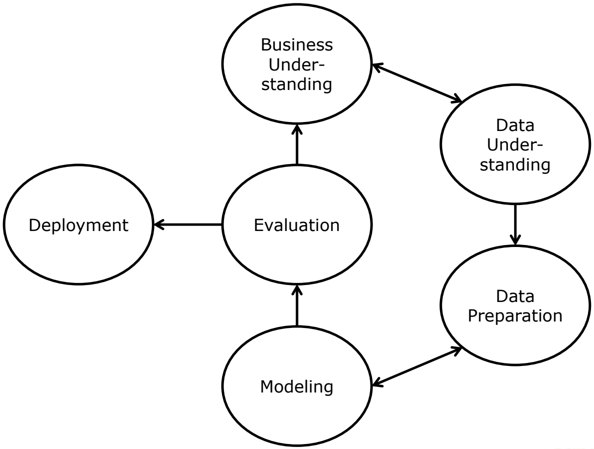
Research Cycle

The following is a representation of the *Research Cycle* used for empirical research in most of the sciences.



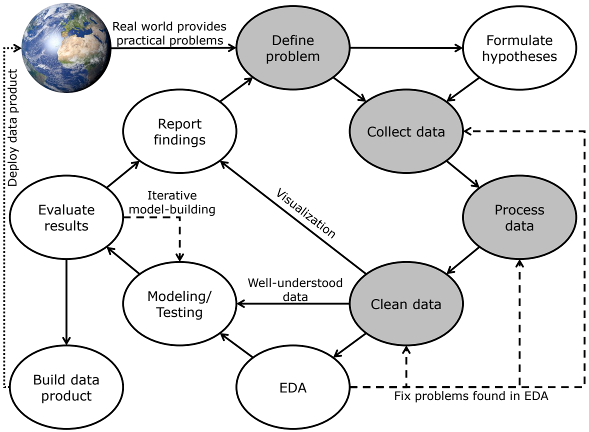
CRISP-DM

The *Cross-industry Standard Process for Data Mining* was developed to standardized the process of data mining in industry applications.



Data Science Cycle

The *Data Science Cycle* represented here was adapted from O'Neil and Schutt (2014).



Data Cleaning

When we receive new data, they are generally messy and contaminated by various anomalies and errors.

- One of the first steps in processing a new set of data is *cleaning*.
- By cleaning the data, we ensure a few properties:
  - The data are in an analyzable format.
  - All data take legal values.
  - Any outliers are located and treated.
  - Any missing data are located and treated.

What are Missing Data?

Missing data are empty cells in a dataset where there should be observed values.

- The missing cells correspond to true population values, but we haven't observed those values.

What are Missing Data?

Missing data are empty cells in a dataset where there should be observed values.

- The missing cells correspond to true population values, but we haven't observed those values.

Not every empty cell is a missing datum.

- Quality-of-life ratings for dead patients in a mortality study
- Firm profitability after the company goes out of business
- Self-reported severity of menstrual cramping for men
- Empty blocks of data following "gateway" items

Missing Data Descriptives

Missing Data Pattern

Missing data (or response) patterns represent unique combinations of observed and missing items.

- P items $\Rightarrow 2^P$ possible patterns.

	X	Y
1	x	y
2	x	.
3	.	y
4	.	.

Patterns for $P = 2$

	X	Y	Z
1	x	y	z
2	x	y	.
3	x	.	z
4	.	y	z
5	x	.	.
6	.	.	z
7	.	y	.
8	.	.	.

Patterns for $P = 3$

Nonresponse Rates

Percent/Proportion Missing

- The proportion of cells containing missing data
- Should be computed for each variable, not for the entire dataset

Attrition Rate

- The proportion of participants that drop-out of a study at each measurement occasion

Percent/Proportion of Complete Cases

- The proportion of observations with no missing data
- Often reported but nearly useless quantity

Covariance Coverage

- The proportion of cases available to estimate a given pairwise relationship (e.g., a covariance between two variables)
- Very important to have adequate coverage for the parameters you want to estimate

Example

We can calculate basic response rates with simple base R commands.

```
## Load some example data:
data(boys, package = "mice")

## Compute variable-wise proportions missing:
mMat  <- is.na(boys)
mMat %>% colMeans() %>% round(3)

  age   hgt   wgt   bmi    hc   gen   phb    tv   reg
0.000 0.027 0.005 0.028 0.061 0.672 0.672 0.698 0.004
```

Example

```
## Compute observation-wise proportions missing:
pmRow <- rowMeans(mMat)

## Summarize the above:
range(pmRow)

[1] 0.0000000 0.7777778

range(pmRow[pmRow > 0])

[1] 0.1111111 0.7777778

median(pmRow)

[1] 0.3333333

## Compute the proportion of complete cases:
mean(pmRow == 0)

[1] 0.2981283
```

Example

We can use routines from the **mice** package to calculate covariance coverage and response patterns.

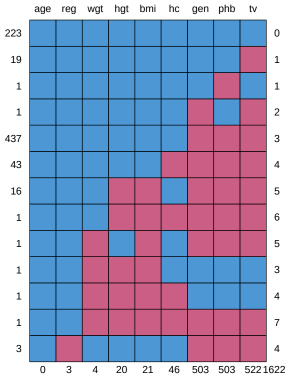
```
## Compute the covariance coverage:
cc <- mice::md.pairs(boys)$rr / nrow(boys)

## Check the result:
round(cc, 2)

     age  hgt  wgt  bmi   hc  gen  phb  tv  reg
age 1.00 0.97 0.99 0.97 0.94 0.33 0.33 0.3 1.00
hgt 0.97 0.97 0.97 0.97 0.92 0.32 0.32 0.3 0.97
wgt 0.99 0.97 0.99 0.97 0.94 0.32 0.32 0.3 0.99
bmi 0.97 0.97 0.97 0.97 0.91 0.32 0.32 0.3 0.97
hc  0.94 0.92 0.94 0.91 0.94 0.33 0.33 0.3 0.93
gen 0.33 0.32 0.32 0.32 0.33 0.33 0.33 0.3 0.33
phb 0.33 0.32 0.32 0.32 0.33 0.33 0.33 0.3 0.33
tv  0.30 0.30 0.30 0.30 0.30 0.30 0.30 0.3 0.30
reg 1.00 0.97 0.99 0.97 0.93 0.33 0.33 0.3 1.00
```

Example

```
## Range of coverages:
range(cc)

[1] 0.2994652 1.0000000

range(cc[cc < 1])

[1] 0.2994652 0.9959893

## How many coverages fall below some threshold?
(cc[lower.tri(cc)] < 0.7) %>% sum()

[1] 21
```

Example

```r
## Compute missing data patterns:
pats <- mice::md.pattern(boys)
```



Example

```
pats

    age reg wgt hgt bmi hc gen phb  tv
223   1   1   1   1   1  1   1   1   1    0
19    1   1   1   1   1  1   1   1   0    1
1     1   1   1   1   1  1   1   0   1    1
1     1   1   1   1   1  1   0   1   0    2
437   1   1   1   1   1  1   0   0   0    3
43    1   1   1   1   1  0   0   0   0    4
16    1   1   1   0   0  1   0   0   0    5
1     1   1   1   0   0  0   0   0   0    6
1     1   1   0   1   0  1   0   0   0    5
1     1   1   0   0   0  1   1   1   1    3
1     1   1   0   0   0  0   1   1   1    4
1     1   1   0   0   0  0   0   0   0    7
3     1   0   1   1   1  1   0   0   0    4
      0   3   4  20  21 46 503 503 522 1622
```

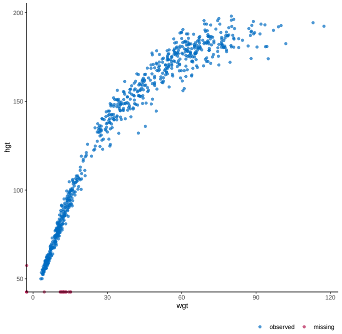
Example

```r
## How many unique response patterns?
nrow(pats) - 1

[1] 13

## What is the most commond response patterns?
maxPat <- rownames(pats) %>% as.numeric() %>% which.max()
pats[maxPat, ]

age reg wgt hgt bmi  hc gen phb  tv
  1   1   1   1   1   1   0   0   0   3
```

Visualizing Incomplete Data

The **ggmice** package provides some nice ways to visualize incomplete data and objects created during missing data treatment.

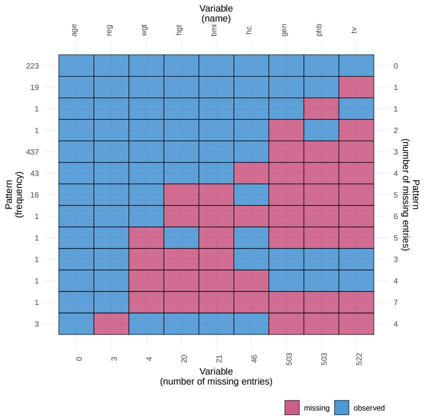
```
library(ggmice); library(ggplot2)

ggmice(boys, aes(wgt, hgt)) + geom_point()
```



Visualizing Incomplete Data

We can also create a nicer version of the response pattern plot.

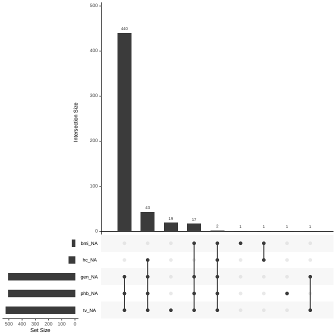
```r
plot_pattern(boys, rotate = TRUE)
```



Visualizing Incomplete Data

The **naniar** package also provides some nice visualization and numerical summary routines for incomplete data.

```
naniar::gg_miss_upset(boys)
```



What is an outlier?

For the time being, we're considering *univariate outliers*.

- Extreme values with respect to the distribution of a variable's other observations
  - A human height measurement of 3 meters
  - A high temperature in Utrecht of 50°
  - Annual income of €250,000 for a student
- Not accounting for any particular model (we'll get to that later)

What is an outlier?

A univariate outlier may, or may not, be an illegal value.

- Data entry errors are probably the most common cause.
- Outliers can also be legal, but extreme, values.

Key Point: We choose to view an outlier as arising from a different population than the one to which we want to generalize our findings.

Finding Univariate Outliers

We have many methods available to diagnose potential outliers.

- Four of the simplest and most popular are:
    1. Internally studentized residuals (AKA Z-score method)
    2. Externally studentized residuals
    3. Median absolute deviation method
    4. Tukey’s boxplot method

Internally Studentized Residuals

For each observation, X_n, we compute the following quantity:

$$T_n = \frac{X_n - \overline{X}}{SD_X}$$

- T_n follows a Student's t distribution with $df = N - 1$.
  - We can do a formal test for “outlier” status.
- Assuming a large sample, if $T_n > C$ (where C is usually 2 or 3), we label X_n as an outlier.

Internally Studentized Residuals

For each observation, X_n, we compute the following quantity:

$$T_n = \frac{X_n - \overline{X}}{SD_X}$$

- T_n follows a Student's t distribution with $df = N - 1$.
  - We can do a formal test for "outlier" status.
- Assuming a large sample, if $T_n > C$ (where C is usually 2 or 3), we label X_n as an outlier.

Although simple, this method has some substantial limitations.

- The cutpoint, C, can only be meaningfully chosen when X is normally distributed.
- Both $\overline{X}$ and SD_X are highly sensitive to outliers.

Externally Studentized Residual

The externally studentized residual method is essentially the same as the internally studentized residual method, but we adjust $\bar{X}$ and SD_X to remove the influence of the observation we're evaluating.

- Let $\mathbb{N}_{(n)} = \{1, \ldots, (n-1), (n+1), \ldots, N\}$.
- Define the deletion mean, $\bar{X}_{(n)}$, and deletion SD, $SD_{X(n)}$, as:

$$\bar{X}_{(n)} = \frac{1}{N-1} \sum_{i \in \mathbb{N}_{(n)}} X_i$$

$$SD_{X(n)} = \sqrt{\frac{1}{N-2} \sum_{i \in \mathbb{N}_{(n)}} \left(X_i - \bar{X}_{(n)} \right)^2}$$

Externally Studentized Residual

The externally studentized residual is defined in the same way as the internally studentized version:

$$T_{(n)} = \frac{X_n - \overline{X}_{(n)}}{SD_{X(n)}}$$

- $T_{(n)}$ follows a Student's t distribution with $df = N - 2$.
  - We can do a formal test for "outlier" status.
- Assuming a large sample, if $T_{(n)} > C$ (where C is usually 2 or 3), we label X_n as an outlier.

Externally Studentized Residual

The externally studentized residual is defined in the same way as the internally studentized version:

$$T_{(n)} = \frac{X_n - \overline{X}_{(n)}}{SD_{X(n)}}$$

- $T_{(n)}$ follows a Student's t distribution with $df = N - 2$.
  - We can do a formal test for "outlier" status.
- Assuming a large sample, if $T_{(n)} > C$ (where C is usually 2 or 3), we label X_n as an outlier.

$T_{(n)}$ is immune to the influence of the nth observation.

- Still requires X to be normally distributed
- Still sensitive to outliers other than the nth observation

Median Absolute Deviation Method

The biggest limitation of studentized residuals is that their measures of central tendency and dispersion are sensitive to outliers.

- If we can replace the (deletion) mean and the (deletion) SD with more robust statistics, we can avoid this issue.
  - Replace the mean, $\bar{X}$, with the *median*, $\text{Med}(X)$
  - Replace the SD with the *median absolute deviation*:

$$MAD_X = b \times \text{Med}\left(\left|X_n - \text{Med}(X)\right|\right)$$

  - We choose the coefficient as $b = 1/Q_{0.75}$
  - For the normal distribution, $b \approx 1/0.6745 \approx 1.4826$

Median Absolute Deviation Method

We compute our test statistic by replacing the mean with the median and the SD with the MAD in the standard Wald test formula:

$$T_{MAD} = \frac{X_n - \text{Med}(X)}{MAD_X}$$

- T_{MAD} doesn't allow for formal statistical tests.
- We can use the same general cutoffs we would use for the studentized residual methods.
    - Assuming a large sample, if $T_{(n)} > C$ (where C is usually 2 or 3), we label X_n as an outlier.

Median Absolute Deviation Method

We compute our test statistic by replacing the mean with the median and the SD with the MAD in the standard Wald test formula:

$$T_{MAD} = \frac{X_n - \text{Med}(X)}{MAD_X}$$

- T_{MAD} doesn't allow for formal statistical tests.
- We can use the same general cutoffs we would use for the studentized residual methods.
  - Assuming a large sample, if $T_{(n)} > C$ (where C is usually 2 or 3), we label X_n as an outlier.

T_{MAD} is immune to the influence of, up to, 50% outlying observations.

- Requires us to assume a parametric distribution for X
  - This assumption is necessary to compute b.

Breakdown Point

To compare robust statistics, we consider their *breakdown points*.

- The breakdown point is the minimum proportion of cases that must be replaced by ∞ to cause the value of the statistic to go to ∞.

The mean has a breakdown point of $1/N$.

- Replacing a single value with ∞ will produce an infinite mean.

The deletion mean has a breakdown point of $2/N$.

- We can replace, at most, 1 value with ∞ without producing an infinite mean.

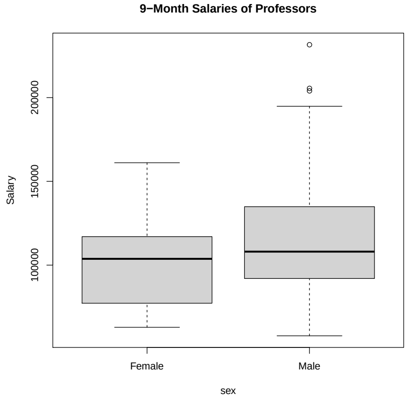
The median has breakdown point of 50%.

- We can replace $n < N/2$ of the observations with ∞ without producing an infinite median.

Boxplot Method

Tukey (1977) described a procedure for flagging potential outliers based on the familiar box-and-whiskers plot.

- Does not require normally distributed X
- Not sensitive to outliers
- Doesn't allow for formal statistical tests



Boxplot Method

A *fence* is an interval defined as the following function of the *first quartile*, the *third quartile*, and the *inner quartile range* ($IQR = Q_3 - Q_1$):

$$F = \{Q_1 - C \times IQR, Q_3 + C \times IQR\}$$

- Taking $C = 1.5$ produces the *inner fence*.
- Taking $C = 3.0$ produces the *outer fence*.

We can use these fences to identify potential outliers:

- Any value that falls outside of the inner fence is a *possible outlier*.
- Any value that falls outside of the outer fence is a *probable outlier*.

Multivariate Outliers

Sometimes, the combinations of values in an observation are very unlikely, even when no individual value is an outlier.

- These observations are *multivariate outliers*.
  - A person in the 95*th* percentile for height and the 5*th* percentile for weight
  - A person who simultaneously scores highly on scales of depression and positive affect

To detect multivariate outliers, we use *distance metrics*.

- Distance metrics quantify the similarity of two vectors.
  - Similarity between two observations
  - Similarity between an observation and the mean vector

Mahalanobis Distance

One of the most common distance metrics is the *Mahalanobis Distance*.

- The Mahalanobis distance, Δ, is a multivariate generalization of the internally studentized residual:

$$\Delta_n = \sqrt{\left(\mathbf{x}_n - \hat{\mu}_{\mathbf{X}}\right) \hat{\Sigma}_{\mathbf{X}}^{-1} \left(\mathbf{x}_n - \hat{\mu}_{\mathbf{X}}\right)^T}$$

As with studentized residuals, if $\Delta_n > C$, we label $\mathbf{x}_n$ as an outlier.

- When $\mathbf{X}$ is K-variate normally distributed, Δ_n^2 follows a χ^2 distribution with $df = K$.
- We take C to be the square-root of a suitably conservative quantile (e.g., $q \in \{99\%, 99.9\%\}$) of the χ_K^2 distribution: $C = \sqrt{\chi_{K,q}^2}$.

Problems with Mahalanobis Distance

Like the internally studentized residual, Mahalanobis distance is highly sensitive to outliers.

- The underlying estimates of central tendency, $\hat{\mu}_{\mathbf{X}}$, and dispersion, $\hat{\Sigma}_{\mathbf{X}}$, are computed using all observations.

Problems with Mahalanobis Distance

Like the internally studentized residual, Mahalanobis distance is highly sensitive to outliers.

- The underlying estimates of central tendency, $\hat{\mu}_{\mathbf{X}}$, and dispersion, $\hat{\Sigma}_{\mathbf{X}}$, are computed using all observations.

We want robust analogues of $\hat{\mu}_{\mathbf{X}}$ and $\hat{\Sigma}_{\mathbf{X}}$.

- We have several options for robust estimation of $\hat{\mu}_{\mathbf{X}}$ and $\hat{\Sigma}_{\mathbf{X}}$. E.g.:
  - Minimum covariance determinant method (MCD; Rousseeuw, 1985)
  - Minimum volume ellipsoid method (MVE; Rousseeuw, 1985)
  - M-estimation (Maronna, 1976)
- Conceptually, robust methods operate by either:
  - Using only a “good” subset of data to estimate $\hat{\mu}_{\mathbf{X}}$ and $\hat{\Sigma}_{\mathbf{X}}$.
  - Downweighting outlying observations when estimating $\hat{\mu}_{\mathbf{X}}$ and $\hat{\Sigma}_{\mathbf{X}}$.

Robust Mahalanobis Distance

Equipped with robust estimates of central tendency, $\hat{\mu}_{R,\mathbf{X}}$, and dispersion, $\hat{\Sigma}_{R,\mathbf{X}}$, we define the robust Mahalanobis distance in the natural way:

$$\Delta_{R,n} = \sqrt{\left(\mathbf{x}_n - \hat{\mu}_{R,\mathbf{X}}\right) \hat{\Sigma}_{R,\mathbf{X}}^{-1} \left(\mathbf{x}_n - \hat{\mu}_{R,\mathbf{X}}\right)^T}$$

We use $\Delta_{R,n}$ in the same way as Δ_n.

- If $\Delta_{R,n} > C$, we label $\mathbf{x}_n$ as an outlier.
- Again, we take C to be the square-root of some quantile of the χ^2_K distribution: $C = \sqrt{\chi^2_{K,q}}$.

Practicalities: Univariate vs. Multivariate

Univariate outlier checks are safe for most variables.

Practicalities: Univariate vs. Multivariate

Univariate outlier checks are safe for most variables.

Don't include too many variables in multivariate outlier checks.

- More variables increases the chances of false positives.
- E.g., don't run a multivariate outlier test on your entire dataset.

Practicalities: Univariate vs. Multivariate

Univariate outlier checks are safe for most variables.

Don’t include too many variables in multivariate outlier checks.

- More variables increases the chances of false positives.
- E.g., don’t run a multivariate outlier test on your entire dataset.

Do use multivariate outlier checks for scales.

- E.g., if you have a psychometric scale measuring depression, you should check the items of that scale for multivariate outliers.

Practicalities: Univariate vs. Multivariate

Univariate outlier checks are safe for most variables.

Don't include too many variables in multivariate outlier checks.

- More variables increases the chances of false positives.
- E.g., don't run a multivariate outlier test on your entire dataset.

Do use multivariate outlier checks for scales.

- E.g., if you have a psychometric scale measuring depression, you should check the items of that scale for multivariate outliers.

Maybe check the variables in a single model for multivariate outliers.

- E.g., if you have a small set of items that you will include in a regression model, it could make sense to check these variables for multivariate outliers.

Practicalities: Outliers for Categorical Data

Nominal, ordinal, and binary items *can* have outliers.

- Outliers on categorical variables are often more indicative of bad variables than outlying cases.

Practicalities: Outliers for Categorical Data

Nominal, ordinal, and binary items *can* have outliers.

- Outliers on categorical variables are often more indicative of bad variables than outlying cases.

Ordinal

- Most participant endorse one of the lowest categories on an ordinal item, but a few participants endorse the highest category.
- The participants who endorse the highest category may be outliers.

Practicalities: Outliers for Categorical Data

Nominal, ordinal, and binary items *can* have outliers.

- Outliers on categorical variables are often more indicative of bad variables than outlying cases.

Ordinal

- Most participant endorse one of the lowest categories on an ordinal item, but a few participants endorse the highest category.
- The participants who endorse the highest category may be outliers.

Nominal

- Groups with very low membership may be outliers on nominal grouping variables.

Practicalities: Outliers for Categorical Data

Nominal, ordinal, and binary items *can* have outliers.

- Outliers on categorical variables are often more indicative of bad variables than outlying cases.

Ordinal

- Most participant endorse one of the lowest categories on an ordinal item, but a few participants endorse the highest category.
- The participants who endorse the highest category may be outliers.

Nominal

- Groups with very low membership may be outliers on nominal grouping variables.

Binary

- If most endorse the item, the few who do not may be outliers.

Treating Outliers

If we locate any outliers, they must be treated.

- Outliers cause by errors, mistakes, or malfunctions (i.e., *error outliers*) should be directly corrected.
- Labeling non-error outliers is a subjective task.
  - A (non-error) outlier must originate from a population separate from the one we care about.
  - Don’t blindly automate the decision process.

Treating Outliers

If we locate any outliers, they must be treated.

- Outliers cause by errors, mistakes, or malfunctions (i.e., *error outliers*) should be directly corrected.
- Labeling non-error outliers is a subjective task.
  - A (non-error) outlier must originate from a population separate from the one we care about.
  - Don’t blindly automate the decision process.

The most direct solution is to delete any outlying observation.

- If you delete non-error outliers, the analysis should be reported twice: with outliers and without.

Treating Outliers

For univariate outliers, we can use less extreme types of deletion.

- Delete outlying values (but not the entire observation).
- These empty cells then become missing data.

Winsorization:

- Replace the missing values with the nearest non-outlying value.

Missing data analysis:

- Treat the missing values along with any naturally-occurring nonresponse.

Treating Outliers

We can also use robust regression procedures to estimate the model directly in the presence of outliers.

- Weight the objective function to reduce the impact of outliers
  - M-estimation
- Trim outlying observations during estimation
  - Least trimmed squares, MCD, MVE
- Take the median, instead of the mean, of the squared residuals
  - Least median of squares
- Model some quantile of the DV’s distribution instead of the mean
  - Quantile regression
- Model the outcome with a heavy-tailed distribution
  - Laplacian, Student’s T

References

Maronna, R. A. (1976). Robust m-estimators of multivariate location and scatter. *The Annals of Statistics*, *4*(1), 51–67. doi: 10.1214/aos/1176343347

O'Neil, C., & Schutt, R. (2014). *Doing data science: Straight talk from the frontline*. Sebastopol, CA: O'Reilly Media, Inc.

Rousseeuw, P. J. (1985). Multivariate estimation with high breakdown point. In W. Grossmann, G. Pflug, I. Vincze, & W. Wertz (Eds.), *Mathematical statistics and applications* (Vol. B, pp. 283–297). Netherlands: Reidel.

Tukey, J. W. (1977). *Exploratory data analysis* (Vol. 2). Reading, MA: Addison–Wesley.