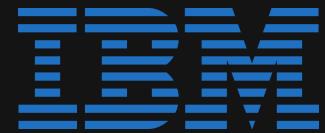
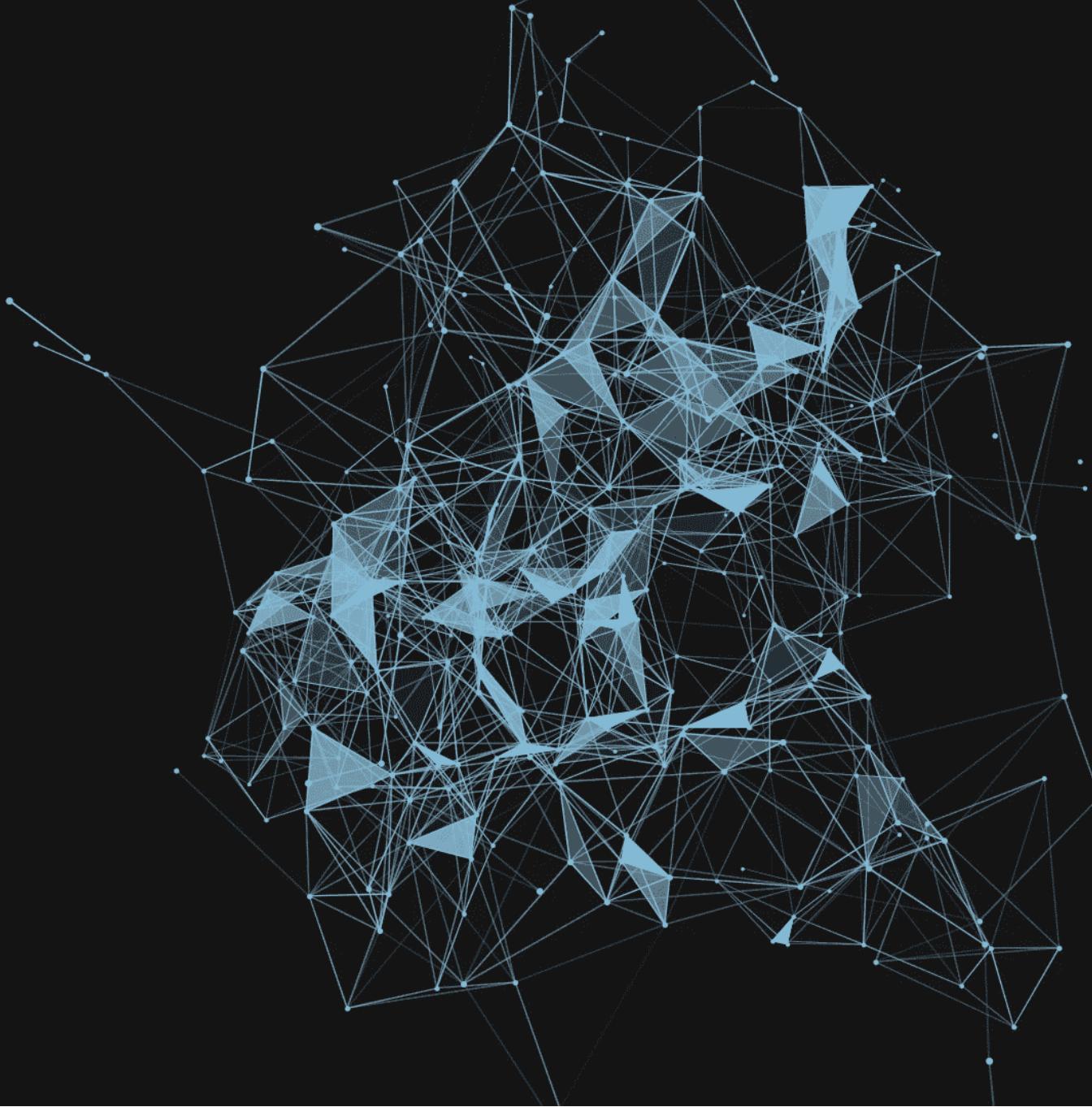


# R + Analytics

Análise de dados utilizando R



Thiago Pires | 14 e 15 Abr 2021  
[github.com/th1460/r-analytics](https://github.com/th1460/r-analytics)



# Analytics

*Decisões baseadas em fatos tornaram-se nossa força competitiva. Usar ou não usar analytics deixou de ser uma opção*

# Analytics

*Analytics é um campo abrangente e multidimensional que se utiliza de técnicas matemáticas, estatísticas, de modelagem preditiva e machine learning para encontrar padrões e conhecimento significativos em dados.*

**Florence Nightingale** registrando e analisando dados de mortalidade na década de 1850. Ela apresentou os dados sobre os soldados feridos com as contagens de mortes por mês. Acabou constatando que a principal **incidência de mortes se devia a infecção hospitalar e não a ferimentos em batalha**, como muitos imaginavam.



A análise de dados pode revelar correlações e padrões. Há menos necessidade de confiar em suposições ou na intuição. E isso pode ajudar a responder a perguntas como:

- O que aconteceu?
- Como ou por que aconteceu?
- O que está acontecendo agora?
- O que provavelmente irá acontecer em seguida?

R

# Linguagem R

Desenvolvida por **Ross Ihaka** e por **Robert Gentleman** no departamento de Estatística da Universidade de Auckland, Nova Zelândia em 1993.



## Exemplo

```
1 + 2 + (3 * 4)
```

```
[1] 15
```

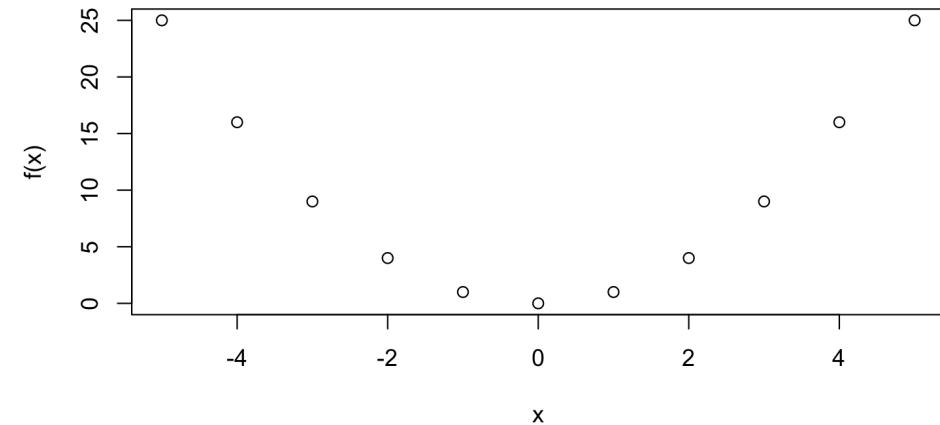
```
f <- function(x) {x^2}
x <- seq(-5, 5, 1); f(x)
```

```
[1] 25 16 9 4 1 0 1 4 9 16 25
```

```
paste("Se x = 2, então f(2) =", f(2))
```

```
[1] "Se x = 2, então f(2) = 4"
```

```
plot(x, f(x))
```



# Base language × Tidyverse

## Base language

```
# sum elements in a vector  
sum(c(10, 5, 8, 12, 2, 0))
```

```
[1] 37
```

```
# data.frame  
df <- data.frame(x = 1:3, y = letters[1:3])  
df$z <- df$x^2
```

```
# filter and select  
df[df$y == "b", c("x", "z")]
```

```
x z  
2 2 4
```

## Tidyverse

```
require(dplyr)  
# sum elements in a vector  
c(10, 5, 8, 12, 2, 0) %>% sum
```

```
[1] 37
```

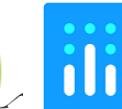
```
# tibble  
df <- tibble(x = 1:3, y = letters[1:3], z = x^2)
```

```
# filter and select  
df %>% filter(y == "b") %>% select(x, z)
```

```
# A tibble: 1 x 2  
      x     z  
   <int> <dbl>  
1     2     4
```

# Packages



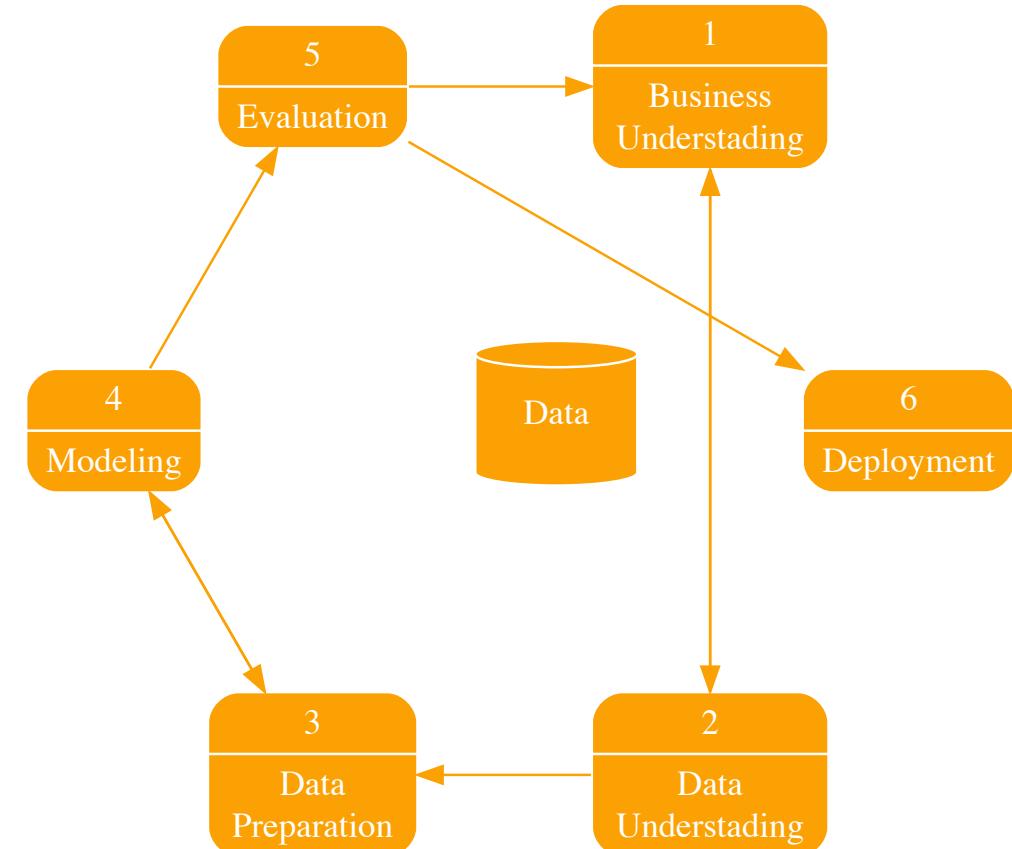
Leaflet  plotly 

# CRISP-DM

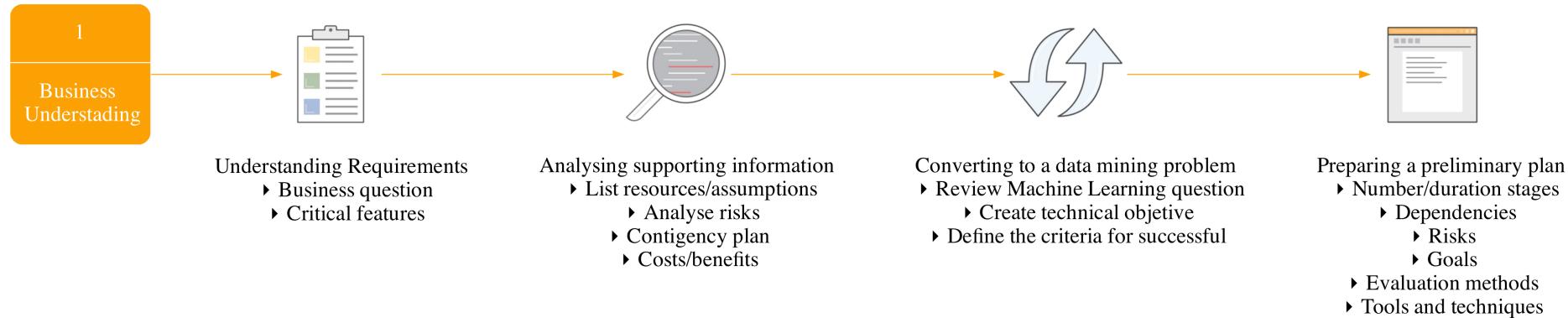
| *Cross Industry Standard Process for Data Mining*

# CRISP-DM

*Cross-industry standard process for data mining, known as CRISP-DM is an open standard process model that describes common approaches used by data mining experts. It is the most widely-used analytics model*



# Business Understanding



## Case Titanic dataset challenge

This is a *part* (train data) from the dataset used in Kaggle Titanic challenge

### Business question:

*What factors was associated with a person survive in the Titanic disaster?*

### Features:

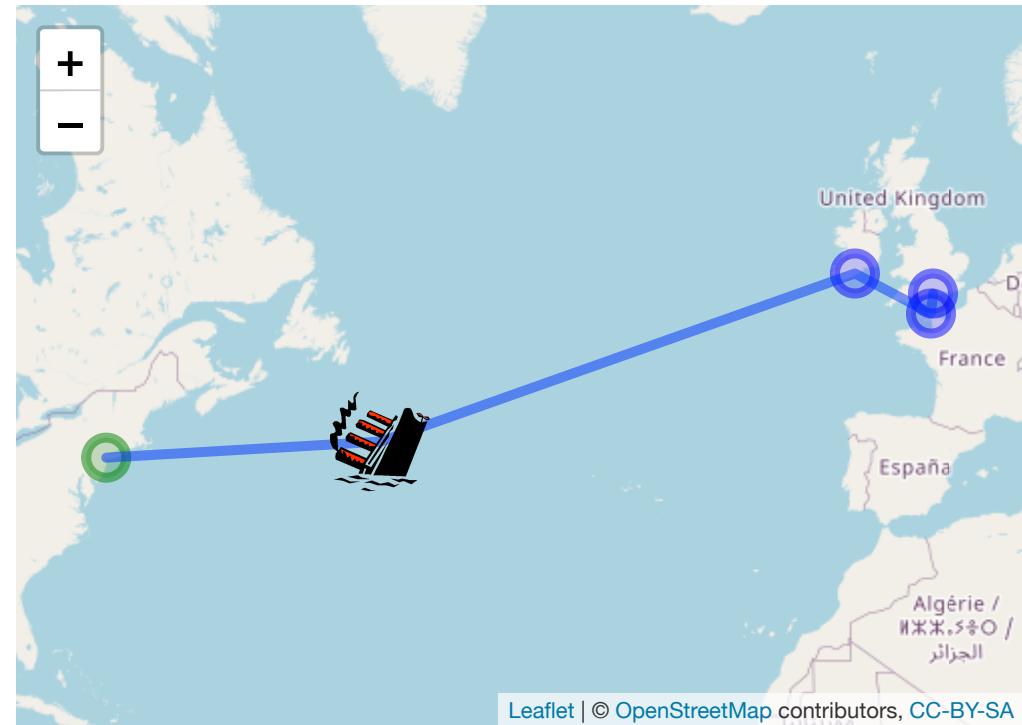
- How to analyze the data? (e.g. R, Python, etc)
- How prepare the data? (e.g. Pandas, dplyr, etc)
- What were the passenger types (e.g. Ages, Gender, Class, etc)

# Titanic disaster

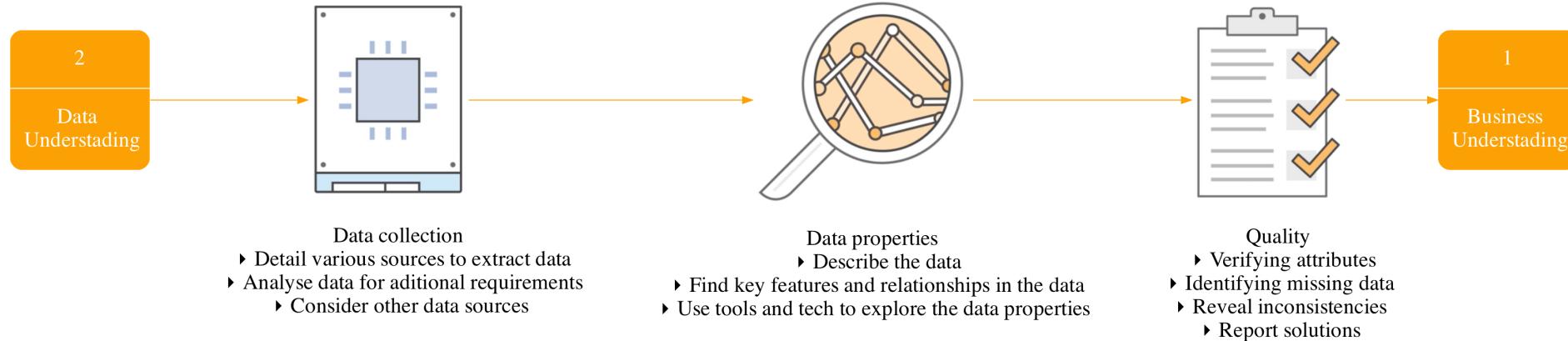
```
require(leaflet)

events <- bind_rows(
  tibble(location = "Southampton (10-04-1912)",
         lng = -1.4191, lat = 50.7894),
  tibble(location = "Cherbourg (10-04-1912)",
         lng = -1.6109, lat = 49.6445),
  tibble(location = "Queenstown (11-04-1912)",
         lng = -8.3211, lat = 51.8535),
  tibble(location = "Naufrágio (14-04-1912)",
         lng = -49.9408, lat = 41.7258),
  tibble(location = "New York",
         lng = -73.9655, lat = 40.6832))

leaflet() %>% addTiles() %>%
  addCircleMarkers(data = events %>% slice(1:3, 5),
                  label = ~location,
                  color = c(rep("blue", 3), "green"))
  addMarkers(data = events %>% slice(4),
             icon = list(
               iconUrl = "resources/images/sinking-shi
               iconSize = c(50, 50)),
             label = ~location) %>%
  addPolylines(data = events, ~lng, ~lat)
```



# Data Understanding



## Titanic case

```
# Read data from Github
dataset <- "https://gist.githubusercontent.com/michhar/2dfd2de0d4f8727f873422c5d959fff5/raw/fa71405126017e6a37bea592"
```

# Data Understanding (Titanic case)

Study the data dictionary and

Inspect the dataset

```
titanic <- readr::read_csv(dataset)
titanic %>% glimpse()
```

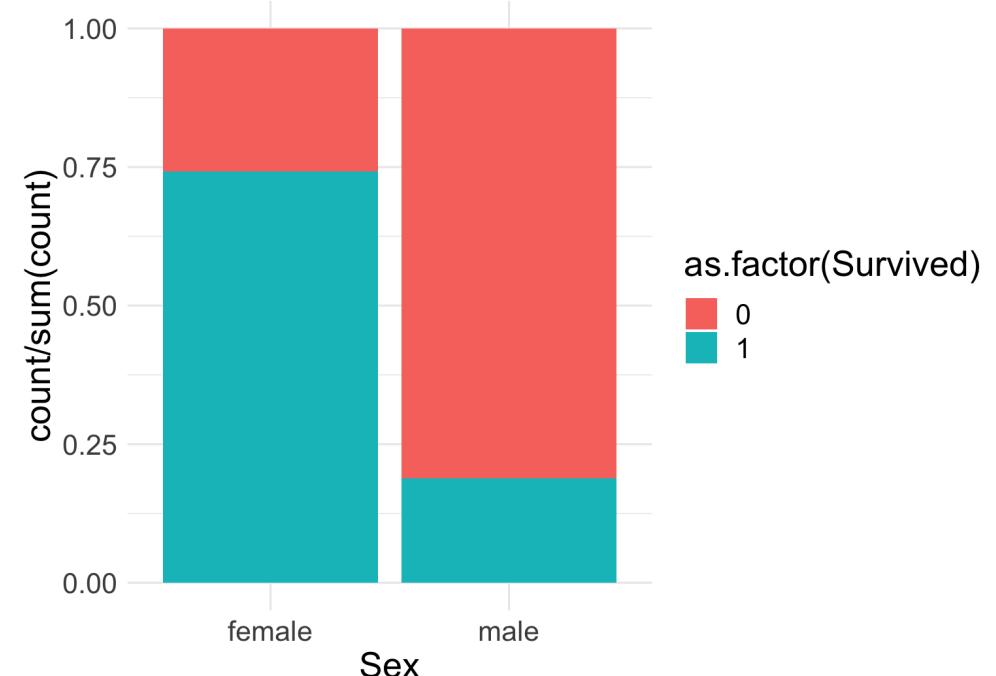
```
Rows: 891
Columns: 12
$ PassengerId <dbl> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, ...
$ Survived      <dbl> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1...
$ Pclass        <dbl> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3, 2, 3, 3...
$ Name          <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bradley (Fl...
$ Sex           <chr> "male", "female", "female", "female", "male", "male", "mal...
$ Age           <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, 39, 14, ...
$ SibSp         <dbl> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4, 0, 1, 0...
$ Parch         <dbl> 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1, 0, 0, 0...
$ Ticket        <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "113803", "37...
$ Fare          <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, 51.8625, ...
$ Cabin         <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA, NA, "G6", "C...
$ Embarked      <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", "S", "S"...
```

# Data Understanding (Titanic case)

## Detect key associations

```
require(ggplot2)
theme_set(
  theme_minimal() +
  theme(text = element_text(size = 18))
)

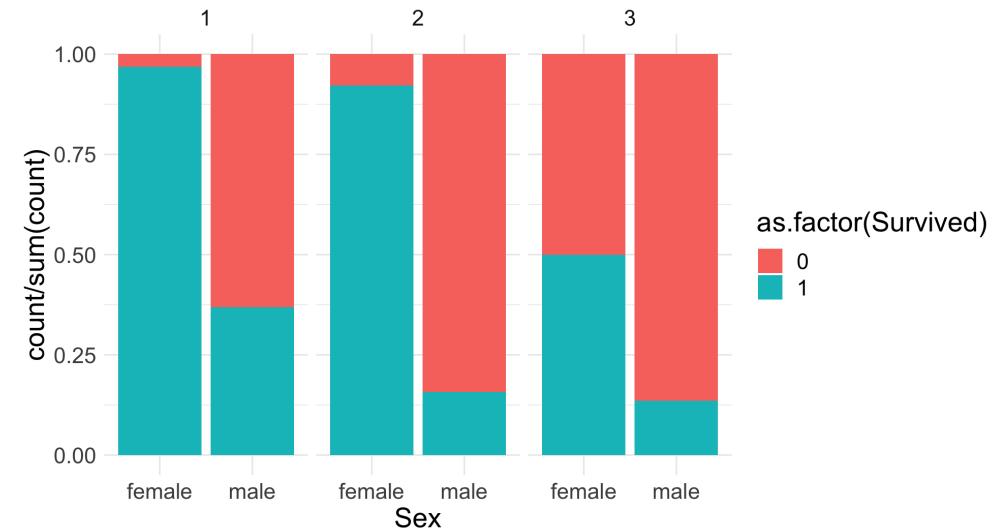
titanic %>%
  ggplot(aes(Sex, ..count../sum(..count..),
             group = Survived,
             fill = as.factor(Survived))) +
  geom_bar(position="fill")
```



# Data Understanding (Titanic case)

## Detect key associations

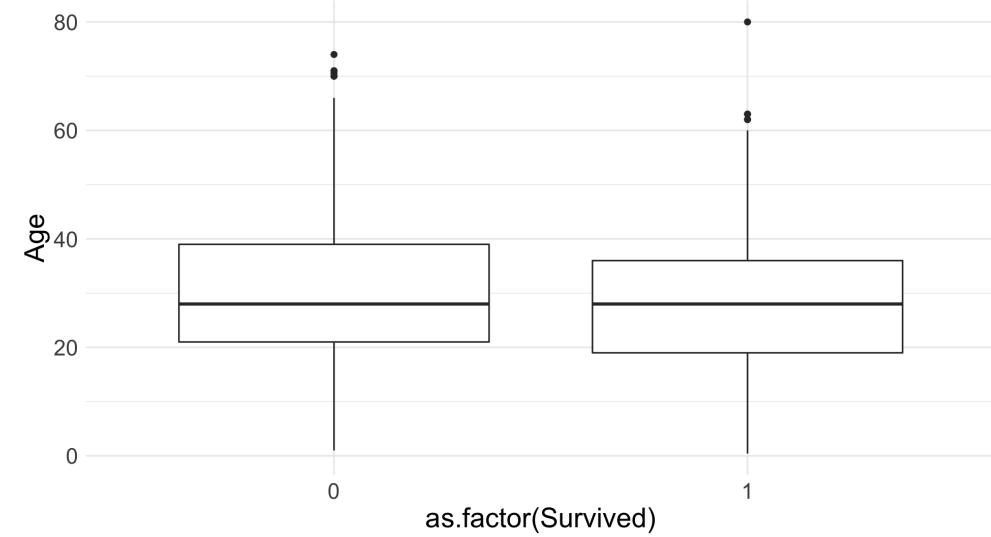
```
titanic %>%
  ggplot(aes(Sex, ..count../sum(..count..),
             group = Survived,
             fill = as.factor(Survived))) +
  geom_bar(position="fill") +
  facet_grid(~Pclass)
```



# Data Understanding (Titanic case)

## Detect key associations

```
titanic %>%
  ggplot(aes(as.factor(Survived), Age)) +
  geom_boxplot()
```



# Data Understanding (Titanic case)

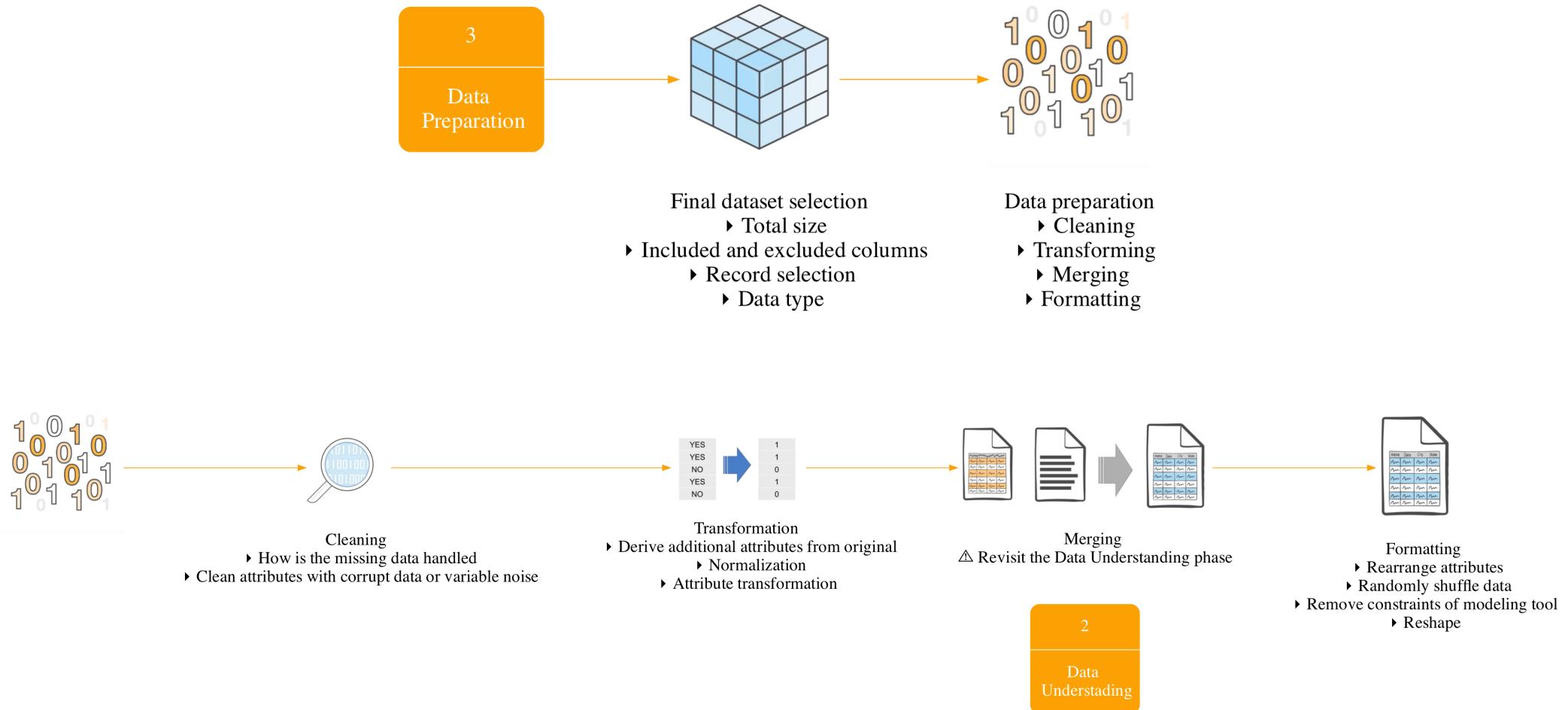
## Identify missing data

```
na_count <- function(x) sum(is.na(x))

titanic %>%
  summarise(across(everything(),
    list(na_count),
    .names = "{.col}")) %>%
  tidyr::pivot_longer(everything(),
    values_to = "n_missing")
```

	# A tibble: 12 x 2
	name n_missing
	<chr> <int>
1	PassengerId 0
2	Survived 0
3	Pclass 0
4	Name 0
5	Sex 0
6	Age 177
7	SibSp 0
8	Parch 0
9	Ticket 0
10	Fare 0
11	Cabin 687
12	Embarked 2

# Data Preparation



# Data Preparation (Titanic case)

Do we drop rows?  
Which?

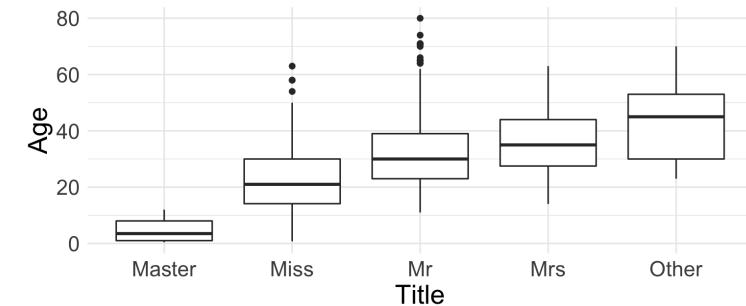
How treat empty  
values?

```
# A tibble: 12 x 2
  name      n_missing
  <chr>        <int>
1 PassengerId     0
2 Survived        0
3 Pclass          0
4 Name            0
5 Sex             0
6 Age           177
7 SibSp          0
8 Parch          0
9 Ticket          0
10 Fare           0
11 Cabin         687
12 Embarked       2
```

```
titanic$name[4]
```

```
[1] "Futrelle, Mrs. Jacques Heath (Lily May Peel)"
```

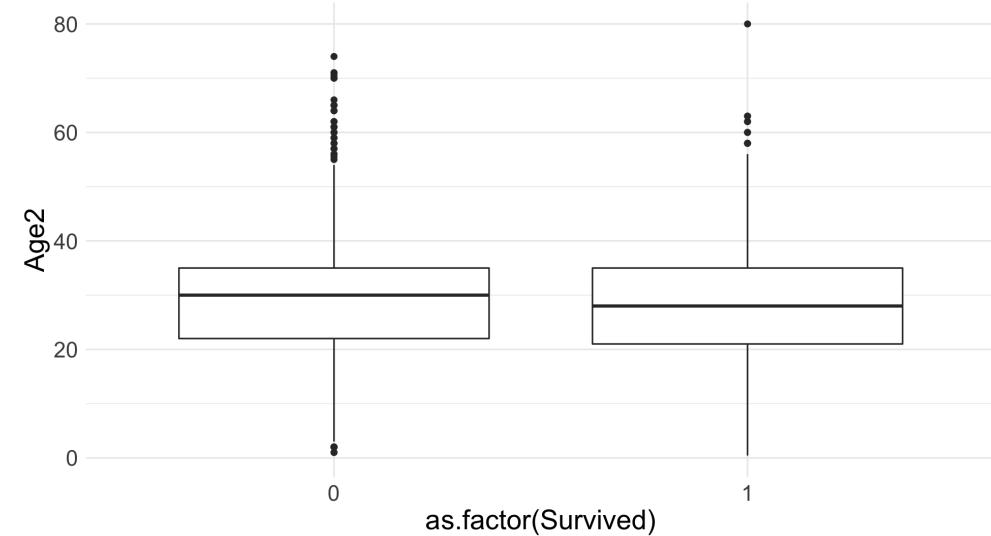
```
require(forcats)
require(ggplot2)
titanic %>%
  mutate(
    Title = Name %>%
      stringr::str_extract(
        "(?=<\\,\\s)(.*)(?=\\.)") %>%
      fct_lump(n = 4)) %>%
  ggplot(aes>Title, Age)) + geom_boxplot
```



# Data Preparation (Titanic case)

## After fix empty ages

```
titanic %>%
  mutate(
    Title = Name %>%
      stringr::str_extract("(?=<\\,\\s)(.*)(?=\\.)") %
      fct_lump(n = 4)) %>%
  group_by(Title) %>%
  mutate(Age2 = ifelse(is.na(Age), median(Age, na.rm =
  ggplot(aes(as.factor(Survived), Age2)) +
  geom_boxplot()
```



# Data Preparation (Titanic case)

## Fix types and labels

```
require(magrittr)
titanic %<%>
  mutate(Sex = as.factor(Sex),
         Pclass = factor(Pclass,
                           labels = c("1st", "2nd", "3rd")),
         Survived = factor(Survived,
                           labels = c("No", "Yes")),
         Embarked = fct_recode(Embarked,
                               "Cherbourg" = "C",
                               "Queenstown" = "Q",
                               "Southampton" = "S"))
titanic %>% select(Sex, Survived, Embarked) %>%
  print(n = 4)
```

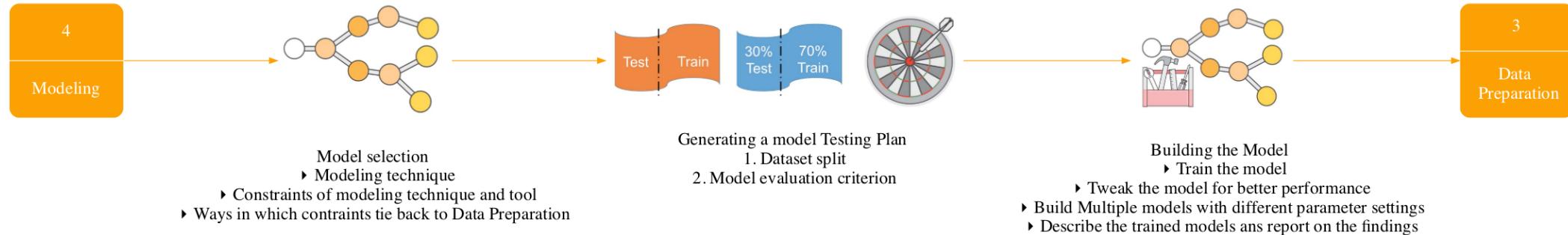
```
# A tibble: 891 x 3
  Sex   Survived Embarked
  <fct> <fct>   <fct>
1 male   No      Southampton
2 female Yes     Cherbourg
3 female Yes     Southampton
4 female Yes     Southampton
# ... with 887 more rows
```

```
freq <- function(data, x) {
  table <- data %>%
    count({{x}}) %>%
    mutate(`%` = (n/sum(n) * 100) %>% round(1)) %>%
    rename(Levels = 1, N = 2)
  tibble(Variable = quo_name(quo({{x}}))) %>%
    bind_cols(table)
}

c("Sex", "Survived", "Embarked", "Pclass") %>%
  purrr::map_dfr(~ freq(titanic, !! sym(.x)))
```

	Variable	Levels	N	`%`
	<chr>	<fct>	<int>	<dbl>
1	Sex	female	314	35.2
2	Sex	male	577	64.8
3	Survived	No	549	61.6
4	Survived	Yes	342	38.4
5	Embarked	Cherbourg	168	18.9
6	Embarked	Queenstown	77	8.6
7	Embarked	Southampton	644	72.3
8	Embarked	NA	2	0.2
9	Pclass	1st	216	24.2
10	Pclass	2nd	184	20.7
11	Pclass	3rd	491	55.1

# Modeling



## Titanic case

*What factors was associated with a person survive in the Titanic disaster?*

### Model selection

- Logistic regression, Decision Tree, etc
- Features: Sex, Pclass, interaction between Sex and Pclass?
- Split strategy

# Modeling (Titanic case)

## Split data

```
require(tidymodels)
set.seed(555)

# Put 3/4 of the data into the training set
data_split <- initial_split(titanic, prop = 3/4)

# Create data frames for the two sets:
train_data <- training(data_split)
test_data <- testing(data_split)
```

# Modeling (Titanic case)

## Survived ~ Sex + Pclass

```
lr_mod <-
  logistic_reg() %>%
  set_engine("glm")

lr_fit1 <-
  lr_mod %>%
  fit(Survived ~ Sex + Pclass, data = train_data)

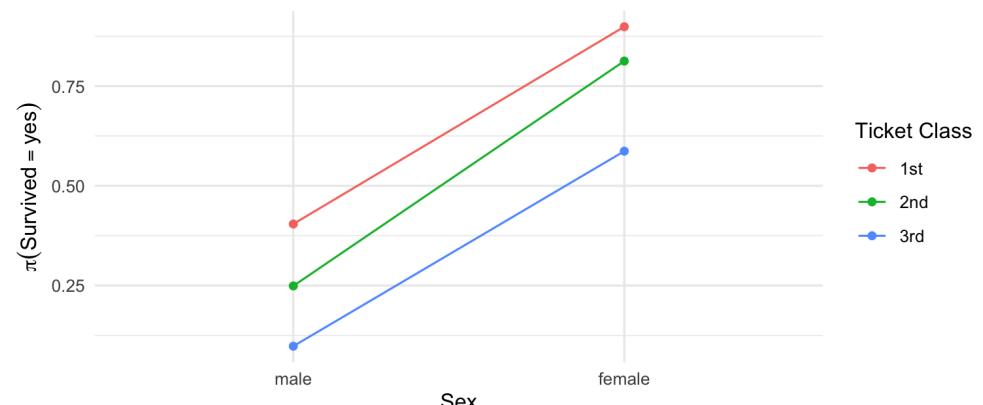
lr_fit1 %>% tidy()
```

```
# A tibble: 4 x 5
  term      estimate std.error statistic p.value
  <chr>     <dbl>    <dbl>     <dbl>   <dbl>
1 (Intercept)  2.18     0.249     8.77 1.76e-18
2 Sexmale     -2.57     0.208    -12.4  4.56e-35
3 Pclass2nd   -0.716    0.278    -2.58 9.97e- 3
4 Pclass3rd   -1.83     0.247    -7.44 1.03e-13
```

```
newdata <-
  expand.grid(Pclass = c("1st", "2nd", "3rd"),
             Sex = c("male", "female"))

pihat <-
  (lr_fit1 %>% predict(newdata, type = "prob")) %>%
  .pred_Yes

newdata %>% mutate(Pihat = pihat) %>%
  ggplot(aes(Sex, Pihat,
             group = Pclass, colour = Pclass)) +
  geom_line() + geom_point() +
  labs(x = "Sex", y = expression(pi(Survived == yes)),
       colour = "Ticket Class") +
  theme_minimal()
```



# Modeling (Titanic case)

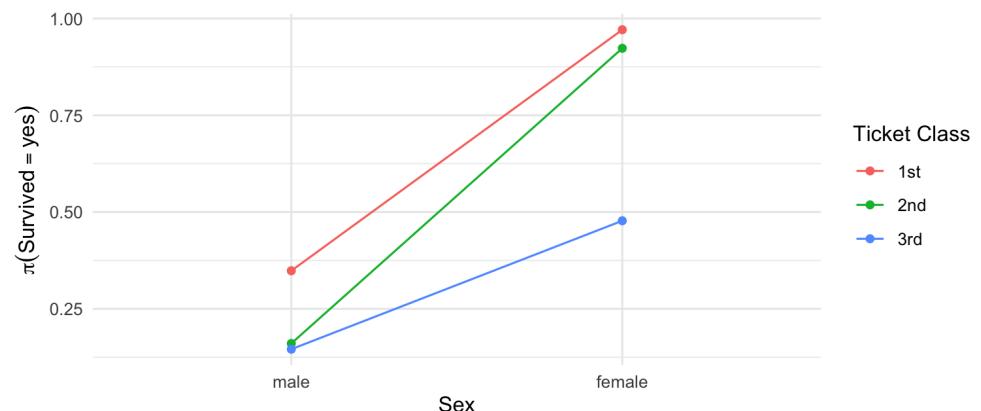
**Survived ~ Sex + Pclass + Sex:Pclass**

```
lr_fit2 <-
  lr_mod %>%
  fit(Survived ~ Sex * Pclass, data = train_data)

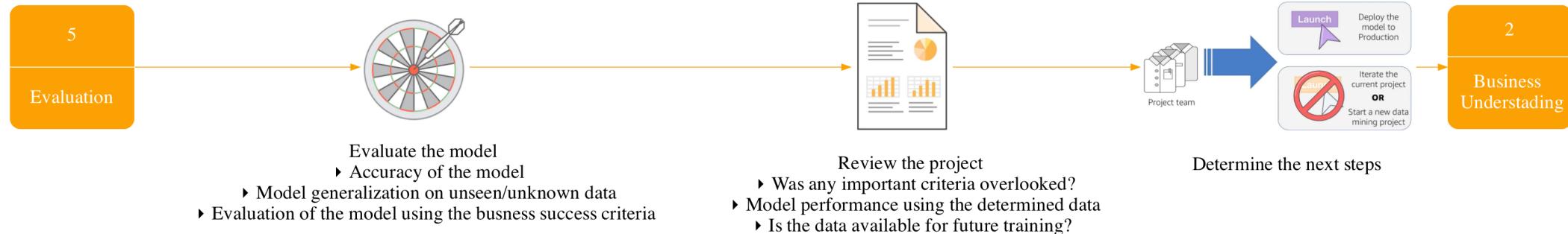
lr_fit2 %>% tidy()
```

	term	estimate	std.error	statistic	p.value
	<chr>	<dbl>	<dbl>	<dbl>	<dbl>
1	(Intercept)	3.51	0.718	4.89	0.000000990
2	Sexmale	-4.14	0.751	-5.51	0.000000363
3	Pclass2nd	-1.03	0.855	-1.20	0.230
4	Pclass3rd	-3.60	0.742	-4.85	0.00000122
5	Sexmale:Pclass2nd	-0.00146	0.934	-0.00157	0.999
6	Sexmale:Pclass3rd	2.46	0.795	3.09	0.00198

```
newdata <-
  expand.grid(Pclass = c("1st", "2nd", "3rd"),
             Sex = c("male", "female"))
pihat <-
  (lr_fit2 %>% predict(newdata, type = "prob")) %%%
  .pred_Yes
newdata %>% mutate(Pihat = pihat) %>%
  ggplot(aes(Sex, Pihat,
             group = Pclass, colour = Pclass)) +
  geom_line() + geom_point() +
  labs(x = "Sex", y = expression(pi(Survived == yes)),
       colour = "Ticket Class") +
  theme_minimal()
```



# Evaluation



## Mensures

Predicted\Actual	Survived	No Survived
Survived	True Positive (TP)	False Positive (FP)
No Survived	False Negative (FN)	True Negative (TN)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$F_{1-score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

# Evaluation (Titanic case)

```
measure <- function(data) {  
  data %>% accuracy(truth = Survived, .pred_class) %>%  
    bind_rows(data %>% f_meas(truth = Survived, .pred_class))  
}
```

Survived ~ Sex + Pclass

```
predict(lr_fit1, test_data) %>%  
  bind_cols(predict(lr_fit1,  
                    test_data, type = "prob")) %>%  
  bind_cols(test_data %>% select(Survived)) %>%  
  measure()
```

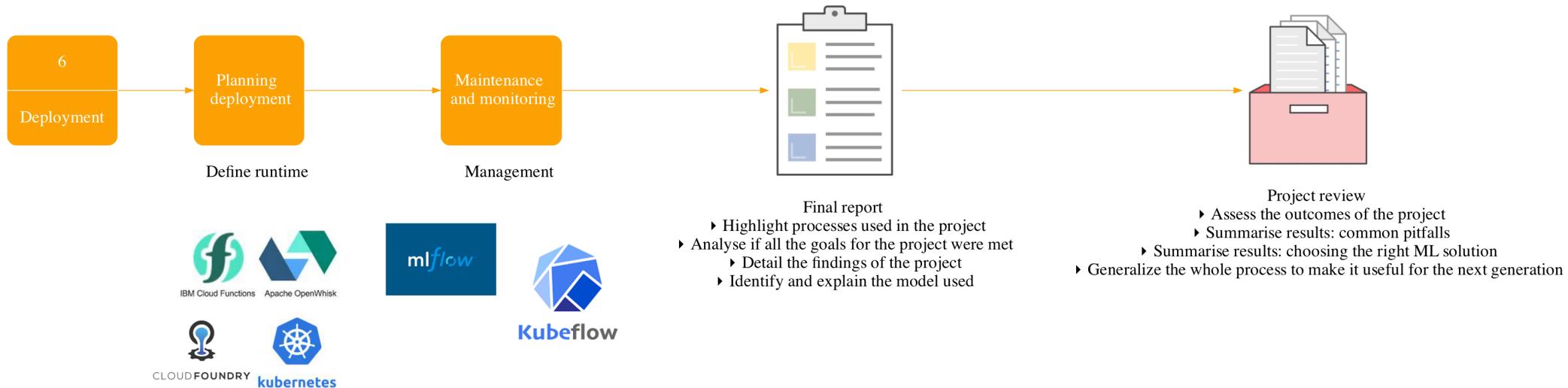
```
# A tibble: 2 x 3  
  .metric  .estimator .estimate  
  <chr>    <chr>        <dbl>  
1 accuracy  binary      0.802  
2 f_meas   binary      0.850
```

Survived ~ Sex + Pclass +  
Sex:Pclass

```
predict(lr_fit2, test_data) %>%  
  bind_cols(predict(lr_fit2,  
                    test_data, type = "prob")) %>%  
  bind_cols(test_data %>% select(Survived)) %>%  
  measure()
```

```
# A tibble: 2 x 3  
  .metric  .estimator .estimate  
  <chr>    <chr>        <dbl>  
1 accuracy  binary      0.779  
2 f_meas   binary      0.850
```

# Deployment



# Deployment (Titanic case)

## Save model

```
yaml::write_yaml(  
  tidypredict::parse_model(lr_fit1),  
  "R/my_model.yml"  
)
```

# Deployment (Titanic case)

## IBM Cloud Functions

IBM Cloud™ Functions service is an event-driven compute platform, also referred to as Serverless computing, or as Function as a Service (FaaS), that runs code in response to events or direct invocations.

## Configure

### exec

```
#!/bin/bash

# run R script
chmod +x script.R # turn executable
echo "$@" > input.json # set input
./script.R # run script
```

### Dockerfile

```
FROM openwhisk/dockerskeleton
RUN apk update && apk add R R-dev R-doc build-base
RUN R -e "install.packages(c('jsonlite', 'tidypredict', 'yaml'), repos = 'http://cran.rstudio.com/')"
```

# Deployment (Titanic case)

## IBM Cloud Functions

### Configure

#### script.R

```
#!/usr/bin/env Rscript

# load model
loaded_model <-
  tidy.predict::as_parsed_model(
    yaml::read_yaml("my_model.yml"))

# input
input <- jsonlite::fromJSON("input.json", flatten = FALSE)

# compute prediction
pred <- tidy.predict::tidy.predict_to_column(as.data.frame(input), loaded_model)

# output
jsonlite::stream_out(pred, verbose = FALSE)
```

# Deployment (Titanic case)

## IBM Cloud Functions

### Deploy

```
# docker
docker build th1460/titanic .
docker push th1460/titanic

# login
ibmcloud login -sso
ibmcloud target --cf

# zip
zip -r titanic.zip exec script.R my_model.yml

# deploy
ibmcloud fn action create titanic titanic.zip --docker th1460/titanic --web true
```

# Deployment (Titanic case)

## IBM Cloud Functions

### Request

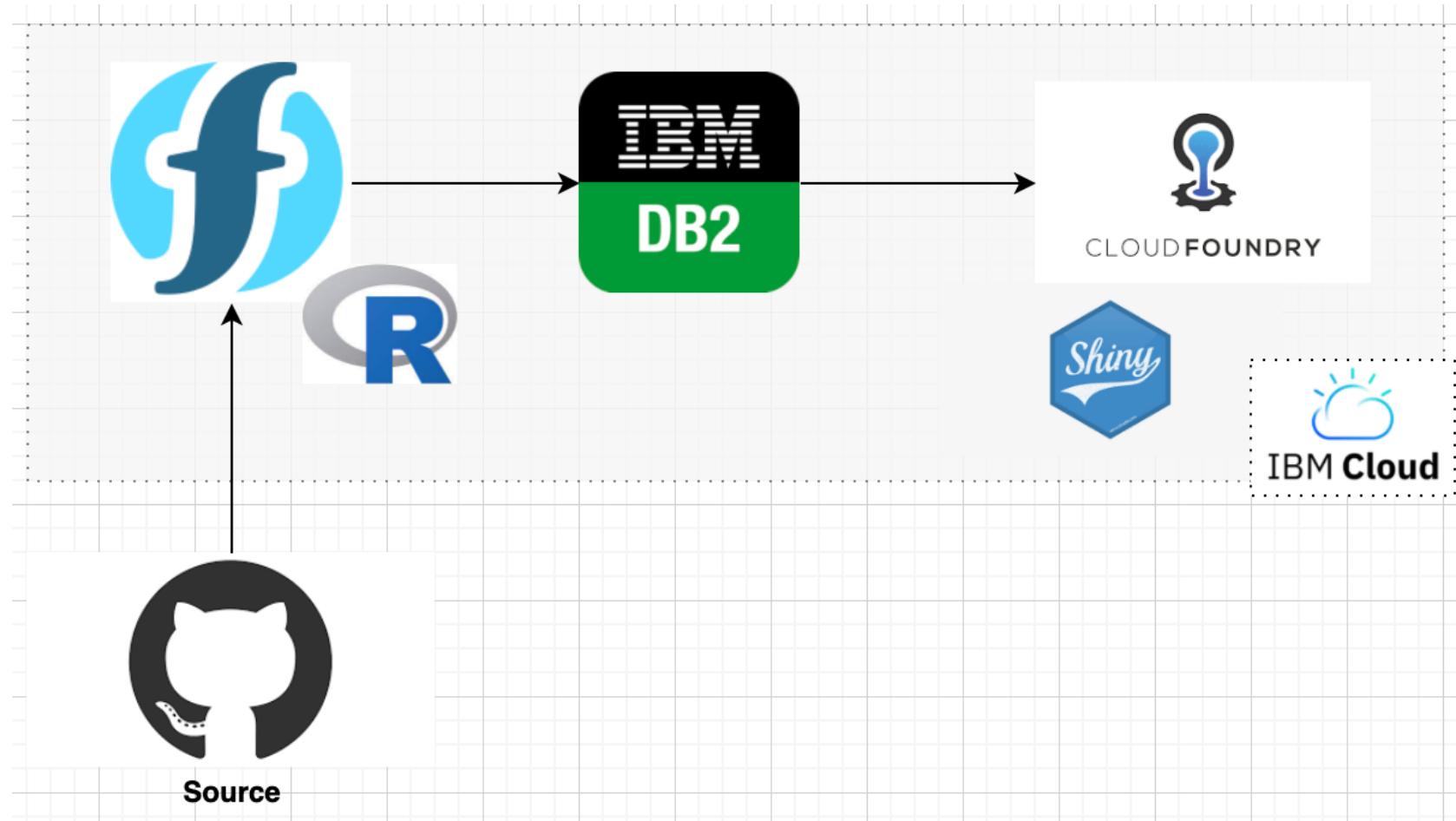
```
input <- list(Sex = "male", Pclass = "3rd")

"https://us-south.functions.appdomain.cloud/api/v1/web/thiago.pires%40ibm.com_dev/default/titanic.json" %>%
  httr::POST(., body = input, encode = "json") %>%
  httr::content() %>%
  .[c("Sex", "Pclass", "fit")] %>%
  jsonlite::toJSON(pretty = TRUE, auto_unbox = TRUE)

{
  "Sex": "male",
  "Pclass": "3rd",
  "fit": 0.0979
}
```

# Covid 19 Analysis

# Covid 19 Analysis



# Covid 19 Analysis

## IBM Functions

- Dockerfile.update
- exec
- read\_update\_data.R
- .Renvironment

```
# Docker build
docker build -t th1460/update-covid -f Dockerfile.update .
```

```
# Docker push
docker push th1460/update-covid
```

```
# Package files
zip -r update_covid.zip exec read_update_data.R .Renvironment
```

```
# Create function
ibmcloud fn action create update-covid19 update_covid.zip --docker th1460/update-covid --web true
```

# Covid 19 Analysis

## Cloud foundry

- Dockerfile.app
- app.R
- manifest.yml

```
# Docker build
docker build -t th1460/dash-covid -f Dockerfile.app .

# Docker push
docker push th1460/dash-covid

# Deploy shiny
ibmcloud cf push dash-covid --docker-image th1460/dash-covid -k 2G
```

# Covid 19 Analysis

## Dashboard



# References

- <https://www.aws.training/Details/eLearning?id=27200>
- <https://the-modeling-agency.com/crisp-dm.pdf>



th1460.github.io  
github.com/th1460  
linkedin.com/in/thop  
slack: @thop