# Data Manipulation in R

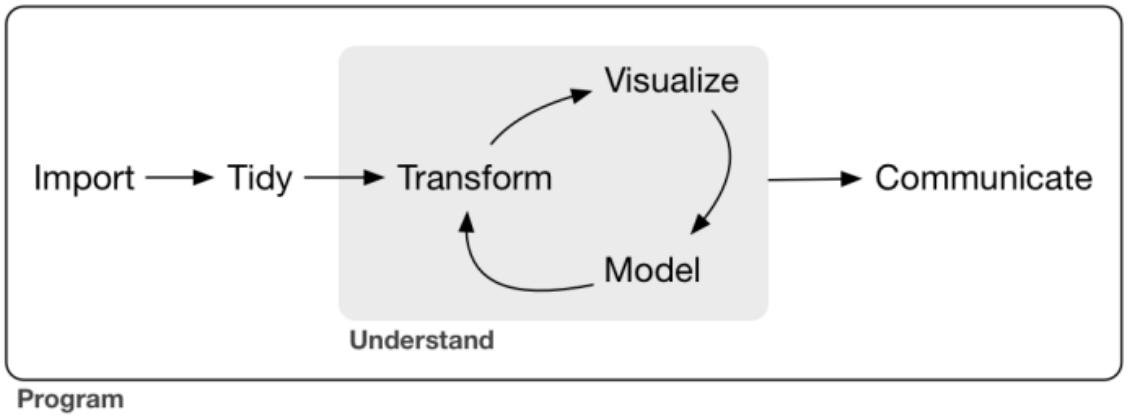
Yuanhao Lai

Western Data Science Solutions

2019-11-07

# Data Analysis

# Workflow


Import
Tidy
Transform
Visualize
Model
Understand
Communicate
Program

# tidyverse

```
library(tidyverse)
```

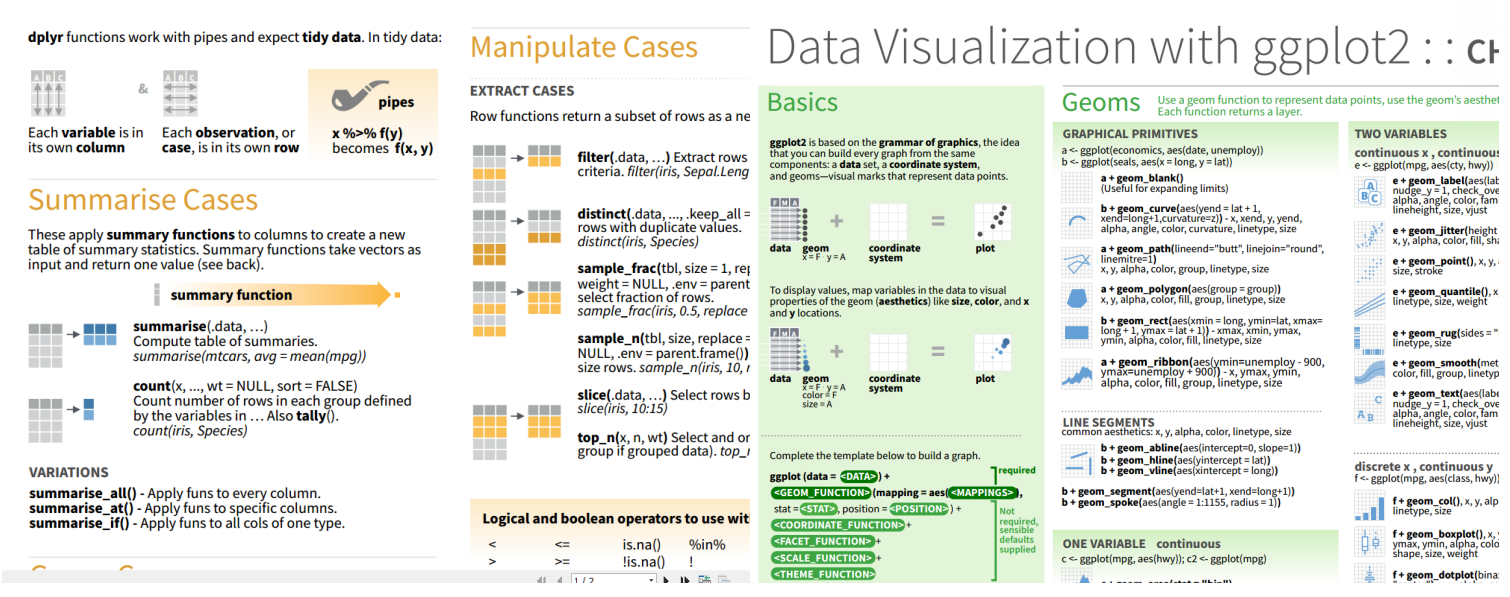
```
## -- Attaching packages --------------------------------------- tidyverse 1.2.1 --

## v ggplot2 3.0.0     v purrr   0.3.2
## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   0.8.2     v stringr 1.4.0
## v readr   1.1.1     v forcats 0.3.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()
```

# tidyverse - The Book

- Overview of data sciences with tidyverse.
- Written by the author of tidyverse!
- Free Ebook online.

# tidyverse - The Cheatsheet

## Data Transformation with dply



**dplyr** functions work with pipes and expect **tidy data**. In tidy data:

Each **variable** is in its own **column**

Each **observation**, or **case**, is in its own **row**

**pipes**

**x %>% f(y)** becomes **f(x, y)**

### Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

summary function

**summarise**(.data, …)
Compute table of summaries.
*summarise(mtcars, avg = mean(mpg))*

**count**(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
*count(iris, Species)*

**VARIATIONS**

**summarise_all()** - Apply funs to every column.
**summarise_at()** - Apply funs to specific columns.
**summarise_if()** - Apply funs to all cols of one type.

### Manipulate Cases

**EXTRACT CASES**

Row functions return a subset of rows as a ne

**filter**(.data, …) Extract rows
criteria. *filter(iris, Sepal.Leng*

**distinct**(.data, ..., .keep_all =
rows with duplicate values.
*distinct(iris, Species)*

**sample_frac**(tbl, size = 1, re
weight = NULL, .env = parent
select fraction of rows.
*sample_frac(iris, 0.5, replace*

**sample_n**(tbl, size, replace =
NULL, .env = parent.frame())
size rows. *sample_n(iris, 10,*

**slice**(.data, …) Select rows b
*slice(iris, 10:15)*

**top_n**(x, n, wt) Select and or
group if grouped data). *top_*

**Logical and boolean operators to use wit**

| | | | |
|---|---|---|---|
| < | <= | is.na() | %in% |
| > | >= | !is.na() | ! |

### Data Visualization with ggplot2 : : cH

#### Basics

**ggplot2** is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.

data + geom (x = F, y = A) + coordinate system = plot

To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

data + geom (x = F, y = A, color = F, size = A) + coordinate system = plot

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                   required
<GEOM_FUNCTION> (mapping = aes( <MAPPINGS> ),
stat = <STAT>, position = <POSITION> ) +                   Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

#### Geoms

Use a geom function to represent data points, use the geom's aesthet
Each function returns a layer.

**GRAPHICAL PRIMITIVES**

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

**a + geom_blank()**
(Useful for expanding limits)

**b + geom_curve**(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

**a + geom_path**(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

**a + geom_polygon**(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

**b + geom_rect**(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

**a + geom_ribbon**(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

**LINE SEGMENTS**
common aesthetics: x, y, alpha, color, linetype, size

**b + geom_abline**(aes(intercept=0, slope=1))
**b + geom_hline**(aes(yintercept = lat))
**b + geom_vline**(aes(xintercept = long))

**b + geom_segment**(aes(yend=lat+1, xend=long+1))
**b + geom_spoke**(aes(angle = 1:1155, radius = 1))

**ONE VARIABLE continuous**

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

**TWO VARIABLES**

**continuous x , continuou**
e <- ggplot(mpg, aes(cty, hwy))

**e + geom_label**(aes(lab
nudge_y = 1, check_ove
alpha, angle, color, fam
lineheight, size, vjust

**e + geom_jitter**(height
x, y, alpha, color, fill, sha

**e + geom_point()**, x, y,
size, stroke

**e + geom_quantile()**, x
linetype, size, weight

**e + geom_rug**(sides = "
linetype, size

**e + geom_smooth**(met
color, fill, group, linetyp

**e + geom_text**(aes(labe
nudge_y = 1, check_ove
alpha, angle, color, fam
lineheight, size, vjust

**discrete x , continuous y**
f <- ggplot(mpg, aes(class, hwy))

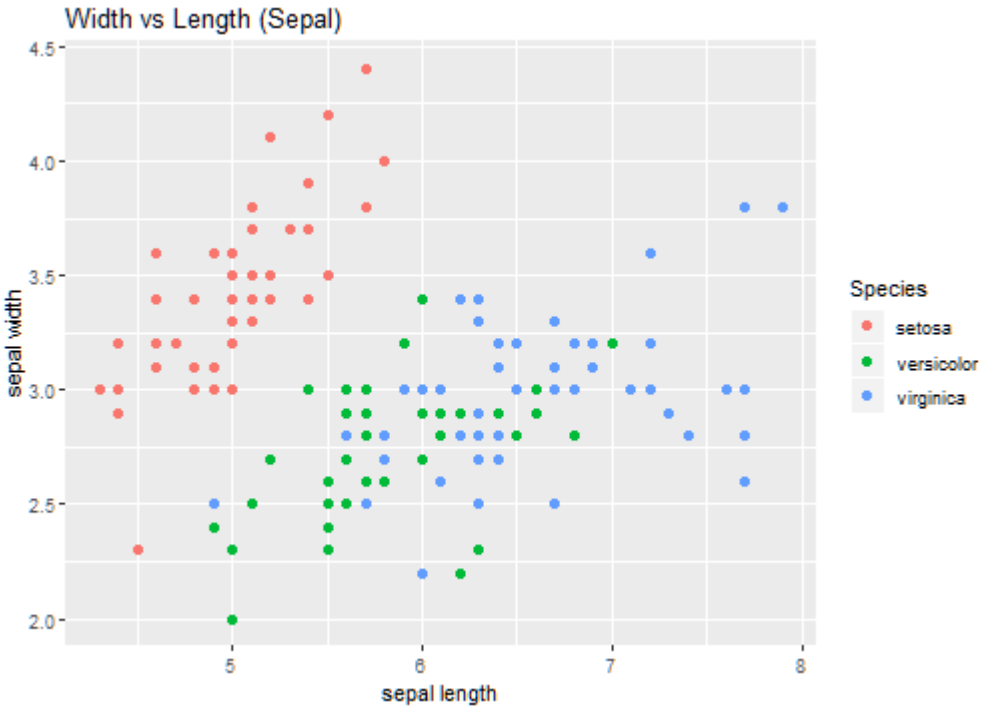
**f + geom_col()**, x, y, alp
linetype, size

**f + geom_boxplot()**, x,
ymax, ymin, alpha, colo
shape, size, weight

**f + geom_dotplot**(bina

# Iris data

This famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris.



# Outline

- Data Transformation
- Data Visualization
- Feature Engineering
- Practise with Titanic

# Data Transformation

# Pipe operator

It is not a must but it makes the code more readable. The functions in Tidyverse were designed to adapt to '%>%'.

```
c(1,2,9,4,5,7,8) %>% mean()
```

```
## [1] 5.142857
```

```
mean(c(1,2,9,4,5,7,8)) #equivalent
```

```
## [1] 5.142857
```

The pipe, '%>%', comes from the package 'magrittr', which is loaded automatically by tidyverse.

# Why dplyr

Compared to the base R data manipulation, `dplyr` is,

- Faster.
- Simple philosophy: data frame in, data frame out.
- Syntax simplicity and ease of use.

# Data Transformation

- as_tibble: an alternative format to data.frame.
- glimpse: get a glimpse of your data.
- select: subset by columns (variables).
- filter: subset rows (observations) meeting one or more conditions.
- arrange: sort data by variables.
- mutate: create new variables.
- summarize: compute summary statistics of data subsets.
- group_by: divide data into groups for later operations.

Most of the above operations have variants with postfix `_if`, `_at`, or `_all`.

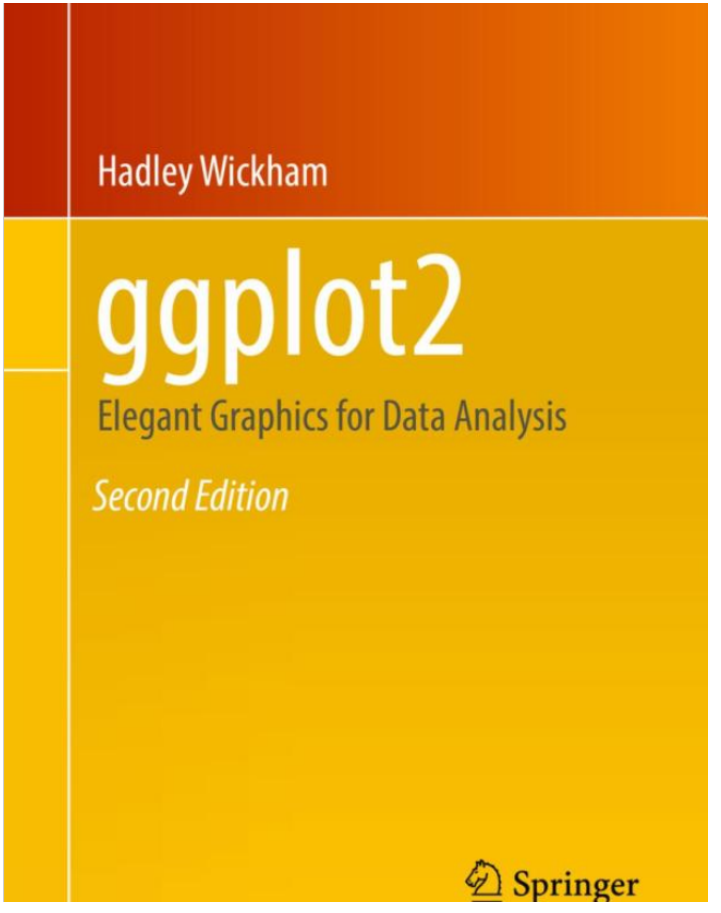
# Iris data

See "dataTransform.Rmd" for examples.

# Data Visualization

# ggplot2

- ggplot2 is a plotting system for R.
- One of the most elegant and most versatile.
- Created by Hadley Wickham.
- Based on the `The Grammar of Graphic`.
- Do more faster by learning one sys- tem and applying it in many places.



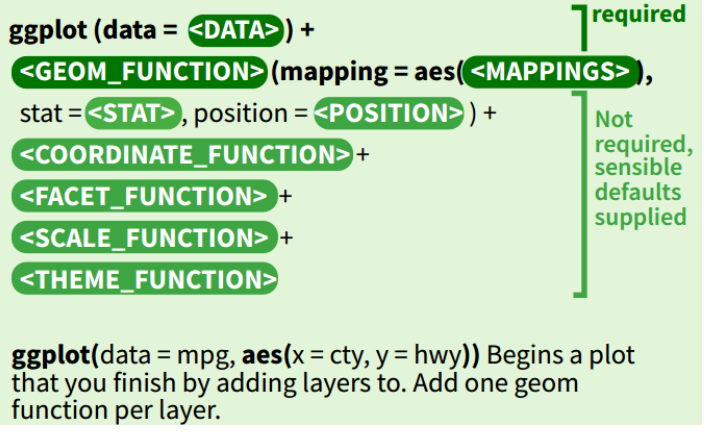
# Why ggplot2?

https://github.com/tidyverse/ggplot2/wiki/Why-use-ggplot2

- Automatic legends, colors, etc.
- The “default” output is much nicer than with base graphics.
- Store any ggplot2 object for modification or future recall.
- Flexibility, intuitiveness, and logic of the mapping between the data and its representation.

# ggplot2 Concepts

- Data and Mapping
- Scale
- Geometric
- Statistics
- Coordinates
- Layer
- Facet



# Data and Mapping

Mapping controls the relationship between variables of data. In ggplot2, we use aesthetic to control the mapping.

| length | width | depth | trt |
|---|---|---|---|
| 2 | 3 | 4 | a |
| 1 | 2 | 1 | a |
| 4 | 5 | 15 | b |
| 9 | 10 | 80 | b |

| x | y | colour |
|---|---|---|
| 2 | 3 | a |
| 1 | 2 | a |
| 4 | 5 | b |
| 9 | 10 | b |

# Scale

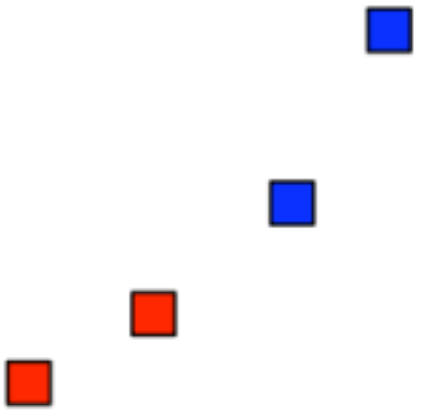
Scale controls the format of the graphical output, related to the mapping.

| x | y | colour |
|---|---|---|
| 2 | 3 | a |
| 1 | 2 | a |
| 4 | 5 | b |
| 9 | 10 | b |

| x | y | colour |
|---|---|---|
| 25 | 11 | red |
| 0 | 0 | red |
| 75 | 53 | blue |
| 200 | 300 | blue |

# Geometric

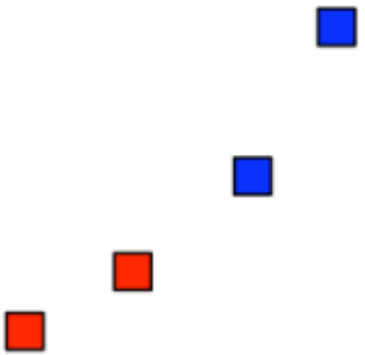
Graphical objects on the plots, e.g., points, lines, polygons.



Geoms

# Statistics

Compute quantitative values/summary statistics of the data and add them to the graph, e.g., the linear regression line, the density.

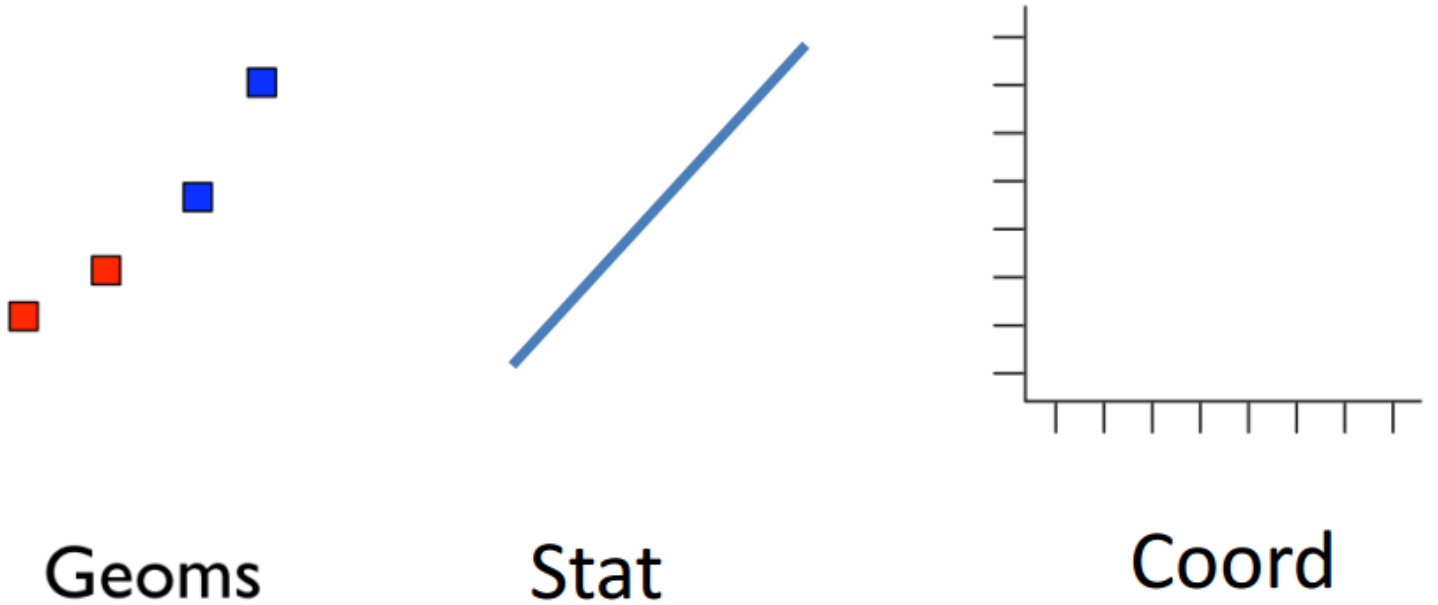


Geoms

Stat

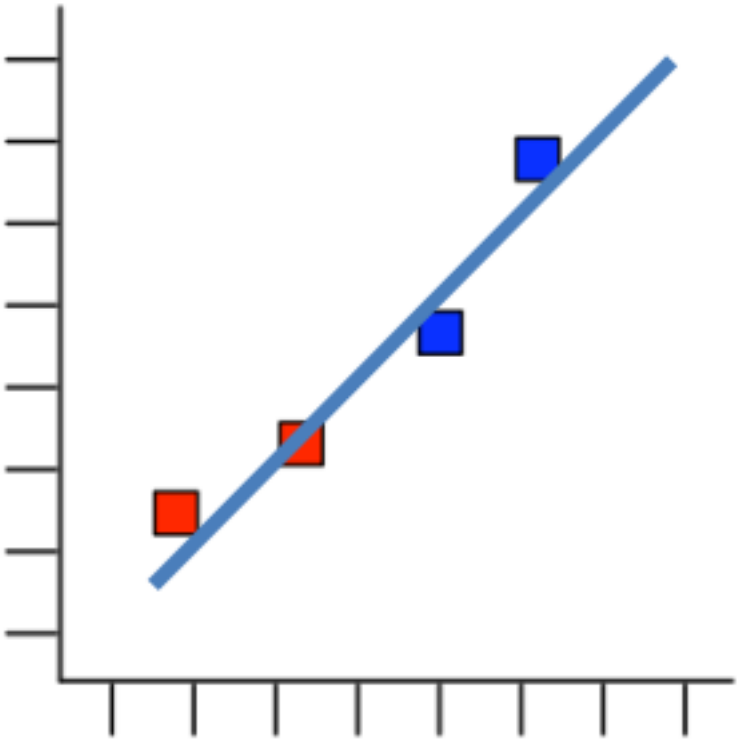
# Coordinates

The visualization perspective, e.g., a grid.



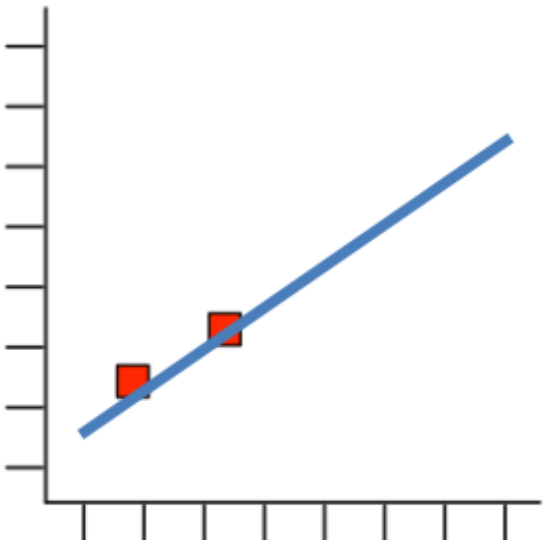
# Layer

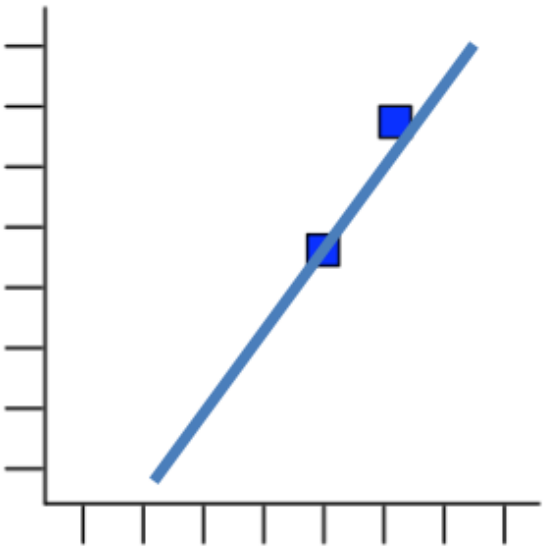
A combination of data, mapping, geometrics, statistics and coordinates, which can be built step by step in ggplot2.



# Facet

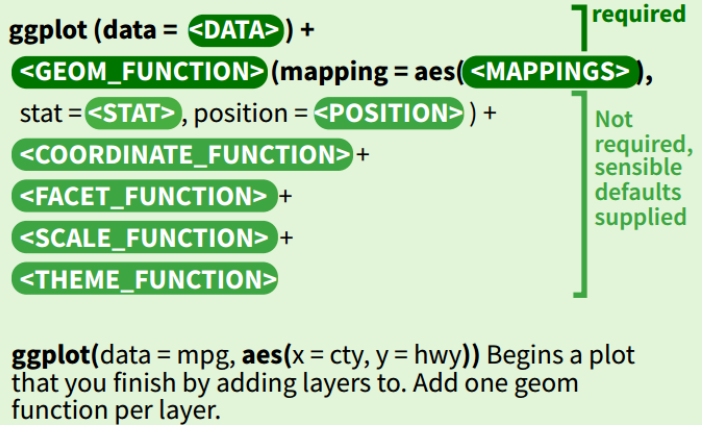
Data are grouped by certain conditions and visualization is done for each group.





# Visualization Types

- Scatterplot
- Bar chart
- Box plot
- Density/Histogram plot
- Pie chart



# Iris data

See "dataVisualize.Rmd" for examples.

# Feature Engineering

Data determine the upper limit of machine learning, and models just approach this upper limit.

Feature engineering helps models spend less effort in learning data.

# Feature engineering

Feature Engineering aims at creating additional relevant features from the existing raw features in the data, and to increase the predictive power of the learning algorithm.

- Most creative aspect of Data Science.
- Hold brainstorming sessions.
- Often require domain knowledges.
- Revist previous experiences.

# Data: Survival in Sinking Titanic

# Survival in Sinking Titanic

https://www.kaggle.com/c/titanic/overview

The sinking of the RMS Titanic is one of the most infamous shipwrecks in history. One of the reasons that the shipwreck led to such loss of life was that there were not enough lifeboats for the passengers and crew. The objective of this data set is to predict which passengers survived the tragedy.

```
## Observations: 891
## Variables: 12
## $ PassengerId <int> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15,...
## $ Survived    <int> 0, 1, 1, 1, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 1, 0,...
## $ Pclass      <int> 3, 1, 3, 1, 3, 3, 1, 3, 3, 2, 3, 1, 3, 3, 3, 2, 3,...
## $ Name        <chr> "Braund, Mr. Owen Harris", "Cumings, Mrs. John Bra...
## $ Sex         <chr> "male", "female", "female", "female", "male", "mal...
## $ Age         <dbl> 22, 38, 26, 35, 35, NA, 54, 2, 27, 14, 4, 58, 20, ...
## $ SibSp       <int> 1, 1, 0, 1, 0, 0, 0, 3, 0, 1, 1, 0, 0, 1, 0, 0, 4,...
## $ Parch       <int> 0, 0, 0, 0, 0, 0, 0, 1, 2, 0, 1, 0, 0, 5, 0, 0, 1,...
## $ Ticket      <chr> "A/5 21171", "PC 17599", "STON/O2. 3101282", "1138...
## $ Fare        <dbl> 7.2500, 71.2833, 7.9250, 53.1000, 8.0500, 8.4583, ...
## $ Cabin       <chr> NA, "C85", NA, "C123", NA, NA, "E46", NA, NA, NA, ...
## $ Embarked    <chr> "S", "C", "S", "S", "S", "Q", "S", "S", "S", "C", ...
```

# Survival in Sinking Titanic

| Variable | Definition | Key |
|---|---|---|
| survival | Survival | 0 = No, 1 = Yes |
| pclass | Ticket class | 1 = 1st, 2 = 2nd, 3 = 3rd |
| sex | Sex | |
| Age | Age in years | |
| sibsp | # of siblings / spouses aboard the Titanic | |
| parch | # of parents / children aboard the Titanic | |
| ticket | Ticket number | |
| fare | Passenger fare | |
| cabin | Cabin number | |
| embarked | Port of Embarkation | C = Cherbourg, Q = Queenstown, S = Southampton |

# Missing rate

```
# Count by summarize_all
train %>%
  summarize_all(list(~sum(is.na(.))/n())) %>%
  gather(key = "variable",value="missingRate")
```

```
## # A tibble: 12 x 2
##    variable    missingRate
##    <chr>             <dbl>
##  1 PassengerId     0
##  2 Survived        0
##  3 Pclass          0
##  4 Name            0
##  5 Sex             0
##  6 Age             0.199
##  7 SibSp           0
##  8 Parch           0
##  9 Ticket          0
## 10 Fare            0
## 11 Cabin           0.771
## 12 Embarked        0.00224
```

# Countinuous/numerical variable

- Binning/discretization, e.g., divide age into categories of children, adult, senior;
- Standarization/Normalization, important in training a neural network;
- Log, minmax transformation, ...

# Binning/discretization

In industry, continuous variables are always discretized first when fitting a logistic regression or a linear regression.

- Improve interpretation sometimes;
- Discretized variables is less sensitive to outliers;
- Discretizing produces more categories/variables and enhances the non-linearity/representation power of the linear model;
- Allow to add more interactions from discretized variables.

# Discretize Age

To discretize a continuous variable in R, we use the function `cut`.

```
summary(train$Age)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.    NA's
##    0.42   20.12   28.00   29.70   38.00   80.00     177
```

```
train <- train %>%
  mutate(age_bin=cut(Age, breaks = c(0,18,60,80),
                     labels=c("children","adult","senior"),
                     include.lowest = TRUE)) %>%
  mutate(age_bin=addNA(age_bin))
train %>% select(Age,age_bin) %>% print(n=2)
```

```
## # A tibble: 891 x 2
##     Age age_bin
##   <dbl> <fct>
## 1    22 adult
## 2    38 adult
## # ... with 889 more rows
```

```
# Categories with NA (not available, i.e., missing values) added
levels(train$age_bin)
```

```
## [1] "children" "adult"    "senior"   NA
```

# Categorical variable

- Always appear in practice.
- High cardinality can create very sparse data.
- Difficult to impute missing.

# Encoding

Mathematical models can not treat categorical variables directly. Instead, categorical variables are encoded before they are passed to the model. Encoding here means that we use some numerical variables to represent a categorical variable.

https://www.slideshare.net/HJvanVeen/feature-engineering-72376750

- One Hot Encoding (Dummy variables)
- Hash encoding
- Label encoding
- Count encoding
- Target encoding
- Category Embedding
- NaN encoding

# One-hot encoding

It is also called dummy variables. For example, to represent the ticket class `Pclass`, which has three levels, 1,2,3, we can create two dummy variables.

```
model.matrix(~factor(Pclass),data=train) %>%
  head()
```

```
##   (Intercept) factor(Pclass)2 factor(Pclass)3
## 1           1               0               1
## 2           1               0               0
## 3           1               0               1
## 4           1               0               0
## 5           1               0               1
## 6           1               0               1
```

# One-hot encoding

- Most basic method without information loss.
- Sparse format is memory-friendly
- Most current implementations don’t gracefully treat missing, unseen variables

# Count encoding

There are 891 obervations in this Titanic data set but only 681 unique tikets. So there were passengers with the same ticket. We can then use the number of passengers for each ticket to represent each ticket.

```
train %>%
  group_by(Ticket) %>%
  summarise(shareTicket=n()) %>%
  right_join(train, by ="Ticket") %>%
  select(Ticket,shareTicket) %>%
  print(n=4)
```

```
## # A tibble: 891 x 2
##   Ticket           shareTicket
##   <chr>                  <int>
## 1 A/5 21171                  1
## 2 PC 17599                   1
## 3 STON/O2. 3101282           1
## 4 113803                     2
## # ... with 887 more rows
```

# Count encoding

In essense, it means that the frequency of a level for a factor is related to the quantity of interest.

- Replace categorical variables with their count in the training set.
- Suitable when a variable has many levels even some of them is rare.
- Useful for both linear and non-linear algorithms.
- Can be sensitive to outliers.
- May add log-transform, works well with counts.
- Replace unseen variables with 1.
- May give collisions: same encoding, different variables.

# Target encoding

Encode categorical variables by their ratio of target (binary classification or regression).

- Be careful to avoid overfit! (Leakage)
- Form of stacking: single-variable model which outputs average target.
- Do in cross-validation manner
- Add random noise to combat overfit
- Suitable when a variable has many levels but each of them is not rare.

# Target encoding

```
reftable <- train %>%
  group_by(Pclass) %>%
  summarise(surPclass=mean(Survived))
reftable
```

```
## # A tibble: 3 x 2
##   Pclass surPclass
##    <int>     <dbl>
## 1      1     0.630
## 2      2     0.473
## 3      3     0.242
```

```
reftable %>%
  right_join(train, by ="Pclass") %>%
  select(Pclass,surPclass) %>%
  mutate(surPclass=surPclass+rnorm(nrow(train),sd=0.01)) %>%  # add noise
  print(n=4)
```

```
## # A tibble: 891 x 2
##   Pclass surPclass
##    <int>     <dbl>
## 1      3     0.248
## 2      1     0.628
## 3      3     0.238
## 4      1     0.638
## # ... with 887 more rows
```

# Exercise

See "Titanic_working.Rmd".

# Thank You!

Please help us make this workshop better

https://forms.gle/qa1gFoTSqFjfGbu2A