VISUALISATION

A PICTURE IS WORTH A THOUSAND WORDS

THE GREATEST VALUE OF A PICTURE IS WHEN IT FORCES US TO NOTICE WHAT WE NEVER EXPECTED TO SEE

JOHN W. TUKEY
(FATHER OF EDA)

SECTION-1

INTRO TO DATA VISUALIZATION AND DISTRIBUTIONS

$x \pm (s) \rightarrow$ std. dev

- Categorical
  - ordinal
  - nominal
- numeric
  - discrete (population)
  - continuous (heights)

- names (heights) # to find out the column names
- ★ discrete numeric data can be considered as ordinal.
- unique (x) # to find all unique values in column
- table (x) # nos of times a variable is occurring in table

variable → x
nos of times → n

new hypothetical use case :- Describe heights to ET (alien)

* to find proportions of male & female

> prop.table (table (heights $ sex))

female	male
0.227	0.773

* cumulative distribution function (CDF)

$F(a) = Pr(x \leq a)$

↓ proportion / probability

⇒ frequency table ⇒ for categorical data

⇒ CDF ⇒ for numeric data

* ECDF → Empirical CDF

* Histograms are prepared after CDF plots

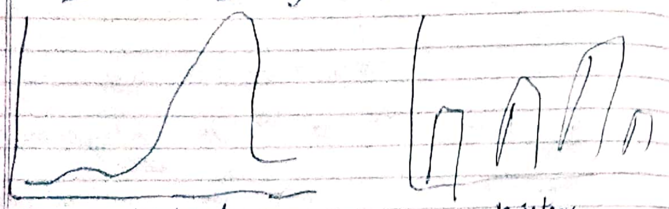
⇒ CDF is essential for calculating probability related to continuous data.

* to represent large data on whole using CDF is not a good idea, we can use CDF to obtain summary, such as probability that a student is b/w 69.5 & 69.5 inches

a <- seq (min (my_data), max(my_data), length = 100)

cdf_function <- function(x) {
  mean (my_data <= x)
}
cdf_values <- sapply (a, cdf_function)
plot (a, cdf_values)

Smooth density plots



smooth density plot   histogram

⇒ degree of smoothness should be taken care of ...

Normal distribution

(gaussian distribution)   (bell curve)

- for any interval a & b, the proportion of values in the interval, is thus using this formula

$$Pr(a < x < b) = \int_a^b \frac{1}{\sqrt{2\pi s}} e^{-\frac{1}{2}\left(\frac{x-m}{s}\right)^2} dx$$

m → mean   s → std deviation
π, e → constants   a, b → intervals

⇒ we have inbuilt function in R to compute this formula

⇒ Its distribution is defined by just 2 parameters, i.e mean & std deviation

★ a low std deviation means, values are closer to mean of the set, while a high std deviation indicates that the values are spread out over a wider range.

★ index <- heights $ sex == "Male"
  x <- heights $ height[index]

average <- mean (x)
SD <- sd (x)
c (average = average, SD = SD)

⇒ average   SD
   69.31   3.61

⇒ standard units (z - score)

- for data which is approximately normal, its convenient to think in terms of standard units. the standard units of value tells us how many standard away from the average this value is

★ example
  z <- scale (x)

⇒ To see how many obs are within 2 sd deviation from the average.

```
mean(abs(z)<2)
[1] 0.95
```

~95% (This is what a normal distribution predicts)

⇒ Z-scores are useful to quickly evaluate whether an observation is average or extreme, Z-scores near 0 are average, Z-score below 2 or above 2 are significantly above or below the mean, and Z-scores above 3 or below -3 are extremely rare

⇒ $\mu \pm \sigma$ ⇒ 68.3% here $|z| \leq 1$

⇒ $\mu \pm 2\sigma$ ⇒ 95.4% here $|z| \leq 2$

⇒ $\mu \pm 3\sigma$ ⇒ 99.7% here $|z| \leq 3$

cumulative distribution for the normal distribution.

pnorm() # function for

F(a) = pnorm(a, avg, s) # distribution for a

⇒ 1 - pnorm(70.5, mean(x), sd(x))
how many students have weight greater than

⇒ for intervals that include exactly one round number

• using data

```
mean(x <= 68.5) - mean(x <= 67.5)
[1] 0.114532
```

• using appx method

```
pnorm(68.5, mean(x), sd(x)) - pnorm(67.5, mean(x), sd(x))
[1] 0.1031677
```

☆ Those data which don't include integers (a proper value are integers) it is not good for using appx (pnorm)

```
mean(x <= 70.9) - mean(x <= 70.1)
[1] 0.07716749
pnorm(70.9, mean(x), sd(x)) - pnorm(70.1, mean(x), sd(x))
[1] 0.0835957
```

⇒ This situation is known as discretization

make QQ-plot with scaled values

observed_quantiles <- quantile(z, p)
theoretical_quantiles <- qnorm(p)
plot(theoretical_quantiles, observed_quantiles)
abline(0,1)

⇒ for normal distribution median & mean are same

☆ Exploratory data analysis. (ex. boxplot)

⇒ distribution of female height

⇒ data visualization helps us to find patterns in our data.

⇒ mad() ⇒ mean absolute deviation

✳ ggplot2

⇒ can be loaded by tidyverse & dplyr
⇒ works with data tables where rows are observations & columns are variables
⇒ grammar of graphics

✳ ggplot2 consists of 3 main components
⇒ Data: The dataset being summarized
⇒ geometry: the type of plot (scatterplot, boxplot ... etc)
⇒ aesthetic mapping: variables mapped to visual axes such as x-axis & y-axis values & color

⇒ There are additional components
- scale
- labels, title, legend
- Theme / style

⇒ p <- ggplot(data = murders)
print(p)

Layers
⇒ geom_symbol

Data %>% ggplot() + layer1 + layer2 + ... + layerN
↓ visualizes geometry

⇒ geom_name(aes(x=, y=), data=, ...) # quick example

⇒ aes # for aesthetic mappings

⇒ ?geom_point for proper explanation of function

⇒ scatterplot

murders %>% ggplot() +
geom_point(aes(x = population/10^6, y = total))

⇒ We can add layers to previously defined objects
p <- ggplot(data = murders)
p + geom_point(aes(population/10^6, total))

geom_label # adds a small rectangular box
geom_text # simply adds a text

p + geom_text(aes(population_10^6, total, label = abb))
↓ abbreviation for state

* tinkering

⇒ call size = 3 (bigger scatter plot now) outside aes.

⇒ nudge_x to move the label a little bit from scatter dot (also out from aes)

⇒ global aes # to give x, y only once in ggplot, not everytime

p <- murders %>% ggplot(aes(population/10^6, total, label = abb))

⇒ now simplified code will be :-

p + geom_point(size = 3) + geom_text(nudge_x = 1

⇒ local mappings to override global ones

p + geom_point(size = 3) + geom_text(aes(x = 10, y = 800, label = "Hello there!"))

Scales, labels and colors

to do the log10 transformation we can use

p + scale_x_continuous(trans = "log10") +
scale_y_continuous(trans = "log10")

more efficient

p + scale_x_log10() + scale_y_log10()

⇒ to add labels

p + xlab("Population in millions (log scale)") +
ylab("Total number of murders (log scale)") +
ggtitle("US gun murders in US 2010")

⇒ avg murder rates

r <- murders %>% summarize(rate = sum(total) / sum(population) * 10^6) %>% pull(rate)

⇒ to change colour according to region

p + geom_point(aes(col = region), size = 3) +

⇒ for adding a straight line

p + geom_abline(intercept = log10(r))

slope is default = 1

⇒ to call line before points & changing lines to dashed lines & to change colour to darkgrey.

p <- p + geom_abline(intercept = log10(3), lty = 2, colour = "darkgrey") + geom_point(aes(col = region), size = 3)

we have called two layers then points (goal)

⇒ to capitalize the legend

p <- p + scale_colour_discrete(name = "Region")

⇒ add-on packages (for more functionality in ggplot)

⇒ library(ggthemes) # to change theme

p + theme_economist()

⇒ to make geom_text as such that labels don't overlap each other

library(ggrepel)

~~p + geom_text_repel()~~

p + geom_text_repel()

other examples

see section 2 → customizing plots → other examples

* dplyr

⇒ summarize & group_by
⇒ dot place holder
⇒ arrange

⇒ summarize returns dataframes

s <- heights %>% filter(sex == "Male") %>% summarize(average = mean(height), standard deviation = sd(height))

⇒ dot placeholder returns numeric

us_murder_rate <- murders %>% summarize(rate = sum(total) / sum(population) * 100000) %>% .$ rate

→ this dot is dot placeholder.

⇒ heights %>% group_by(sex) %>% summarize(average = mean(height), standard_deviation = sd(height))

	sex	average	standard_deviation
1	Female	64.9	3.76
2	Male	69.3	3.61

⇒ arrange for ordering

```
murders %>% arrange(murder_rate) %>% head()
# ascending order
murders %>% arrange(desc(murder_rate)) %>% head()
# descending order
```

⇒ nested sorting (for no ties)

```
murders %>% arrange(region, murder_rate) %>% head
```

sorted by region, then lowest murder rate

⇒ `murders %>% top_n(10, murder_rate)`

top 10 nos of states, but not ordered

⇒ `murders %>% arrange(desc(murder_rate)) %>% top_n(10`

* Gapminder

dslabs

```
library(gapminder)
data(gapminder)
head(gapminder)
```

⇒ country, year, infant_mortality, life_expectancy, fertility, population, gdp, continent, region

- `gapminder %>% filter(year == 2015 & country %in% c("Sri Lanka", "Turkey")) %>% select(country, infant_mortality)`

⇒ scatterplot of life expectancy versus fertility

```
ds_theme_set()
filter(gapminder, year == 1962) %>%
ggplot(aes(fertility, life expectancy)) + geom_point()
```

color = continent

* facetting # side by side plots

```
facet_grid()   # function for this
gapminder %>% filter(gapminder, year %in% c(1962, 2012)) %>%
+ ggplot(aes(fertility, life_expectancy, color = continent)) + geom_point()
+ facet_grid(continent ~ year)
```

rows ↓ columns ↓

facet_grid(. ~ year)

⇒ facet_wrap() # automatically wraps the series of plots, so that resp displays has reasonable dimensions

- changes in previous code

```
years <- c(1962, 1930, 1990, 2000, 2012)
continents <- c("Europe", "Asia")
facet_wrap(~ year)
> filter(year %in% years & continent %in% continents)
```

* Time series plots (time on x-axis)

⇒ single time series

```
gapminder %>% filter(country == "United States") %>%
ggplot(aes(year, fertility)) + geom_line()
```

⇒ multiple times series

```
countries <- c("South Korea", "Germany")
gapminder %> filter(country %in% countries) %>%
ggplot(aes(year, fertility, col = country)) +
  geom_line()
```

⇒ adding text labels

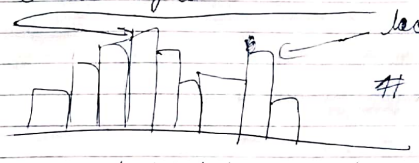
```
labels <- data.frame(country = countries, x = c(1975, 1965), y = c(60, 72))
gapminder %>% filter(country %in% countries) %>%
ggplot(aes(year, life_expectancy, col = country)) + geom_line() +
+ geom_text(data = labels, aes(x, y, label = country), size = 5) +
```

* Transformations (useful for better understanding of distribution)

```
gapminder <- gapminder %>% mutate(dollars_per_day = gdp/population/365)
```

⇒ log transformations

- using base 2 for example means that every time a value doubles, log transformation increases by one



⇒ in normal distribution, mode = average

- use log base 2, log base 10 for EDA

```
past_year <- 1970
gapminder %>%
  filter(year == past_year & !is.na(gdp)) %>%
  ggplot(aes(log2(dollars_per_day))) +
  geom_histogram(binwidth = 1, color = "black")
```

(if gdp is NA, then ignore it)

⇒ To make the distributions scale in normal form and the scale label as log form

gapminder %>%
filter(year == past_year & !is.na(gdp)) %>%
ggplot(aes(dollars_per_day)) +
geom_histogram(binwidth = 1, color = "black") +
scale_x_continuous(trans = "log2")

for proper normal scaling

* Stratify and Boxplot

p <- gapminder %>%
filter(year = past_year & !is.na(gdp)) %>%
mutate(region = reorder(region, dollars_per_day, FUN = median)) %>% # reorder according to means medians
ggplot(aes(region, dollars_per_day, fill = continent)) + # color by continent
geom_boxplot() +
theme(axis.text.x = element_text(angle = 90, hjust = 1)) + # rotate the labels by 90°
xlab("") # to ignore the x axis label
+ scale_y_continuous(trans = "log2") # to properly show the graph

quantiles

quantile (data, q)

Percentile

p <- seq (0.01, 0.99, 0.01)
quantile (data, p)

Quartiles

divide dataset into 4 parts
25% → 1st quartile
50% → median
75% → 3rd quartile

→ summary() returns minimum, quartile & maximum of a vector.

summary (heights $height)
p <- seq (0.01, 0.99, 0.01)
percentiles <- quantile (heights $height, p)

→ 25th & 75th percentile match 1st & 3rd quartile to verify it ....

percentiles [names (percentiles) == "25%"]
percentiles [names (percentiles) == "75%"]

⇒ qnorm() function gives the theoretical value of a quantile with probability p of observing a value equal to or less than quantile value given a normal distribution with mean → mu
std. dev → sigma

qnorm: ~~qnorm~~ qnorm (p, mu, sigma)
qnorm (p)

~~qnorm~~

* quantiles are defined as such that p is the probability of random observation less than or equal to the quantile

* Relation of pnorm() & qnorm()

pnorm (-1.96) ≈ 0.025
⇒ (result of pnorm() is quantile)
qnorm (0.025) ≈ -1.96

⇒ qnorm() & pnorm() are inverse functions

pnorm (qnorm (0.025)) = 0.025

Quantile - Quantile plots

index <- heights $height[sex == "male"]
x <- heights $ height [index]
z <- scale (x)

proportion of data below 69.5
mean (x <= 69.5)

# calculating observed & theoretical quantiles
p <- seq (0.05, 0.95, 0.05)
observed_quantiles <- quantile (x, p)
theoretical_quantiles <- qnorm (p, mean = mean(x), sd = sd(x))

make QQ-plot
plot (theoretical_quantiles, observed_quantiles)