Review of the course "R for Data Science" Part 03(Talk 09~12)

By Haoran Nie @ HUST Life ST

Partically translated by Rui Zhu @ HUST Life ST

双语版

R for bioinformatics, data visualisation

Talk 09

TOC

- basic plot functions
- basic `ggplot2`
- special letters
- equations
- advanced `ggplot2`

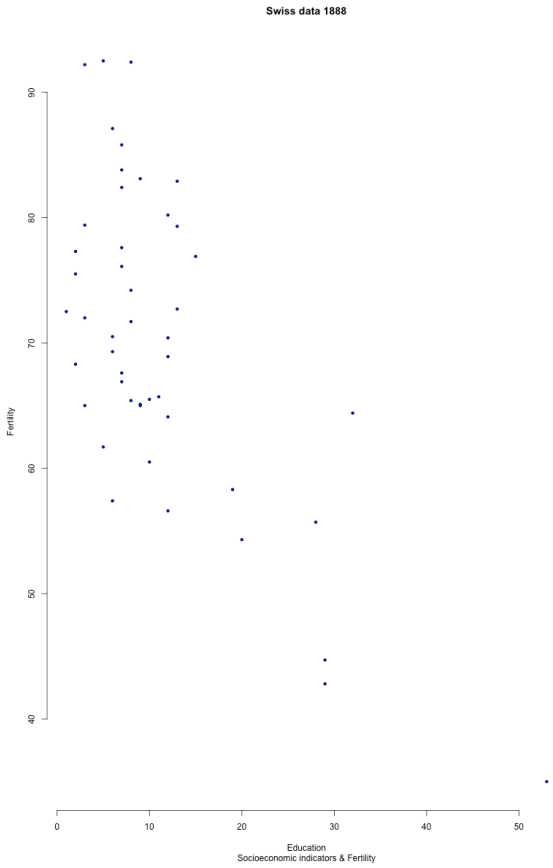
Basic plot functions using R

Dot plot 散点图

An example:

```
with(
  swiss,
  # 将数据框 swiss 中的列作为环境，可以直接使用列名而不需要再加上 swiss$ 前缀。
  plot(
    Education,
    Fertility,
    type = "p",
    main = "Swiss data 1888",
    sub = "Socioeconomic indicators & Fertility",
    xlab = "Education",
```

```
    ylab = "Fertility",
    col = "darkblue",
    xlim = range( Education ),
    ylim = range( Fertility ),
    pch = 20,
    frame.plot = F## 去除图的边框
  )
)
```



Function usage:

```
## Default S3 method:
plot(
  x,
  y,
  type = "p",
  xlim = NULL, ylim = NULL,
  log = "",
  main = NULL, sub = NULL,
  xlab = NULL, ylab = NULL,
  ann = par("ann"), ##par("ann")=T
```

```
  axes = TRUE, frame.plot = axes,
  panel.first = NULL, panel.last = NULL, asp = NA,
  xgap.axis = NA, ygap.axis = NA,
  ...
)
# Default Parameters are listed.
```

Arguments

PARAMETERS	DETAILS
`x`, `y`	the `x` and `y` arguments provide the x and y coordinates for the plot. Any reasonable way of defining the coordinates is acceptable. See the function `xy.coords` for details. If supplied separately, they must be of the same length.
`type`	1-character string giving the type of plot desired. The following values are possible, for details, see `plot`: `"p"` for points, `"l"` for lines, `"b"` for both points and lines, `"c"` for empty points joined by lines, `"o"` for 重叠绘制点和线, `"s"` and `"S"` for stair steps and `"h"` for像柱状图一样的垂直线。 Finally, `"n"` does not produce any points or lines.
`xlim`	the x limits (x1, x2) of the plot. Note that `x1 > x2` is allowed and leads to a '反向轴'.The default value, `NULL`, indicates that the range of the finite values to be plotted should be used.
`ylim`	the y limits of the plot.
`log`	a character string which contains `"x"` if theX轴变为对数的， `"y"` if the Y轴变为对数的 and `"xy"` or `"yx"` if both axes 变为对数的
`main`	a main title for the plot, see also `title`.
`sub`	a subtitle for the plot.
`xlab`	a label for the x axis, defaults to a description of `x`.
`ylab`	a label for the y axis, defaults to a description of `y`.
`ann`	一个逻辑值，指示默认注释（标题以及x轴和y轴标签）是否应显示在图上。
`axes`	逻辑值，指示是否应在图上绘制两个轴。 Use graphical parameter `"xaxt"` or `"yaxt"` to suppress just one of the axes.
`frame.plot`	a logical indicating whether a box should be drawn around the plot.
`panel.first`	要在设定绘图轴之后但在进行任何绘图之前求值的"表达式"。这对于绘制背景网格或散点图平滑非常有用。注意，这是通过惰性求值来实现的：从其他`plot`方法传递这个参数可能不起作用，因为它可能过早地被求值。
`panel.last`	an expression to be evaluated after plotting has taken place but before the axes, title and box are added. See the comments about `panel.first`.
`asp`	Y/X宽高比, see `plot.window`.
`xgap.axis`, `ygap.axis`	the `x/y` axis gap factors, passed as `gap.axis` to the two `axis()` calls (when `axes` is true, as per default).

You can also use `ggplot` to draw the plot above:

```
ggplot(
  swiss,
  aes(x = Education, y = Fertility)
) +
geom_point() +
scale_x_log10() +
scale_y_log10() +
xlab("Education") +
ylab("Fertility") +
ggtitle "Swiss data 1888")
```

High-level and low-level

- **high level**: 绘图函数在图形设备上创建新的绘图
- **low level**: 绘图函数向现有绘图添加更多信息

Low level plots

- `points`：点图
- `lines`：线图
- `abline`：直线
- `polygon`：多边形
- `legend`：图例
- `title`：标题
- `axis`：轴

High level plots

- `plot`：通用画图函数
- `pairs`
- `coplot`
- `qqnorm`
- `hist`
- `dotchart`
- `image`
- `contour`

注：可以用 `add = TRUE` 参数（如果可用）将 high level 函数强制转换为 low level

图形相关参数（系统函数）

`par()` 函数：显示或修改当前**图形设备**的参数。用以下命令查看支持的内容：

```
par( c( "mar", "bg" ) ); ## 显示指定参数的值
## 显示所有参数
par();
```

调整 par() 参数前请备份

par() 用于指定全局参数，因此在改变前尽量备份

```
oldpar <- par(); ## 备份

do some changes here ...

## 恢复
par( oldpar );
```

常用图形参数及调整: margin

图形边距（figure margins ）

分别指定下 -> 左 -> 上 -> 右的边距，即从下面开始，顺时针移动。

```
# 单位是:  text lines
par( mar = c( 5.1, 4.1, 4.1, 2.1 )); ## 设置新 martin
# 单位是:  inch
par( mai = c( 5.1, 4.1, 4.1, 2.1 )); ## 设置新 martin
```

常用图形参数及调整: 多 panel

画 2x3 共 6 个 panel，从左到右。(2 行 3 列)

```
par( mfrow=c(2,3) );
for( i in 1:6 )
  plot( sample( 1:10, 10 ), main = i );
```

画 2x3 共 6 个 panel，从上到下。(2 行 3 列)

```
par( mfcol=c(2,3) );
for( i in 1:6 )
  plot( sample( 1:10, 10 ), main = i );
```

重要概念: 图形设备

图形设备是指图形输出的设备，可以将**图形设备**理解为**保存格式**。

默认设备是:

- X11() : Unix
- windows() : windows

- quartz()：OS X

图形显示在**显示器**上。

图形设备：cont.

常用其它设备有：

- pdf()
- png()
- jpeg()

分别对应输出文件格式。

常用图形设备：pdf()

使用方法如下：

```
pdf( file = "/path/to/dir/<file_name>.pdf", height = 5, width = 5 ); ## 创建一个新设备/ pdf 文件
plot(1:10); ## 作图;
dev.off(); ## 关闭设备
```

说明

1. 默认文件名为 `Rplots.pdf` ，
2. `dev.off()` 必须关闭。关闭后，返回到最近使用的图形设备
3. `height` 和 `width` 参数的单位是 inch
4. 如果运行多个 high level 作图命令，则会产生**多页 pdf**

请尽量使用 pdf 作为文件输出格式

1. 生信图片大多是点线图，适合保存为矢量格式（如 pdf, ps 等）；
2. 矢量图可无限放大而不失真（变成像素）；
3. 可由 Adobe Illustrator 等矢量图软件进行编辑

ggplot2

You should know that

1. `xy -axes` will automatically adjust based on the data you give;
2. `ggplot2` plotting results can be saved in variables and more layers can be added;
3. Layers using their own data need to be specified with `data =`, while global data is not. You can just specify it via `ggplot(data = data.frame(...))`.

Some basic parameters of ggplot2

1. **aes** (aesthetics) 美学：控制全局参数，包括：x,y 轴使用的数据，颜色 (colour, fill) ，形状 (shape)，大小 (size)，分组 (group) 等等；
2. **图层**：`geom_<layer_name>` ；每张图可有多个图层（此处有两个）；图层可使用全局数据 (df) 和参数 (aes)，也可以使用自己的 aes 和数据；

   `geom_<name_of_the_layer>`:

- `geom_point` , `geom_line`: 点线图，用于揭示两组数据间的关系；
- `geom_smooth` : 常与 `geom_point` 联合使用，揭示数据走势
- `geom_bar` ： bar 图
- `geom_boxplot` : 箱线图，用于比较 N 组数据，揭示区别
- `geom_path` : 与 `geom_line` 相似，但也可以画其它复杂图形
- `geom_histogram`, “ geom_density “‘ : 数据的分布，也可用于多组间的比较
- ……

1. `scale` - Display Control 其它控制函数

   `scale_< 控制内容 >_< 控制手段 >`,

   e.g. `scale_color_manual()`: 以**手选方式**控制 **颜色**

- `scale_color_...`
  - `..._gradient()`: 为不同数量的变量及其颜色使用渐变色。
    - `..._gradient2()`
    - `..._gradientn()`
  - `..._brewer`: 使用默认调色板。
- `scale_fill_...`

  - **scale_fill_manual()**：允许用户手动指定每个类别或组的填充颜色。
  - **scale_fill_discrete()**：用于离散变量，自动为每个类别分配不同的颜色。
  - **scale_fill_continuous()**：用于连续变量，根据连续数值分配颜色。
  - **scale_fill_gradient()**、
    scale_fill_gradient2()、
    scale_fill_gradientn()：用于创建渐变颜色填充，适用于表示连续数值范围。

- scale_shape_...
  - **scale_shape_manual()**：允许用户手动指定每个因子水平的形状。
  - **scale_shape_discrete()**：用于离散变量，自动为每个因子水平分配不同的形状。
  - **scale_shape_continuous()**：用于连续变量，根据连续值分配形状（不常用）。

- scale_size_...

fill 与 colour 有什么区别???

- colour 定义几何图形**轮廓**的颜色（形状的“笔划”）
- fill 定义用于**填充**几何图形的颜色
- point 一般只有一个**颜色 没有填充**
- However, point shapes **21–25** that include **both** a colour and a fill.
- colour 在 shape = 21 时，为描边（stroke）色；
- 可用 stroke 控制线条粗细；

调色板和其他包中的相应函数

Included in ggplot2 scale_color_hue, scale_color_manual, scale_color_grey, scale_colour_viridis_d, scale_color_brewer ...

From the RColorBrewer package scale_color_brewer(palette = "<palette name>")

From the viridis package scale_color_viridis(discrete=TRUE, option="<palette name>")

Other packages ...

- `paletteer` package: `scale_color_paletteer_xx` functions
- `ggsci` package
- `ggsci`: 论文发表用的色板!!!
  - contents
    `scale_color_<journal>` 和 `scale_fill_<journal>` functions and color palettes

- supported journals

    * NPG `scale_color_npg()`, `scale_fill_npg()`

    * AAAS, NEJM, Lancet, JAMA ...

1. `size`
2. `shape`

Question:

像 `size`、`colour` 等参数。可以在 `aes()` 里面或外面，有什么区别?

Answer:

在内部时，以指定列的值确定**大小**，或按 factor 的数量确定颜色、形状的**数量**。

放在 `aes()` 外部意味着这些属性与特定的固定值相关联，而不是与数据的某个变量相关联。

Coordinate System 坐标系

1. 线性坐标系

- `coord_cartesian()`,
  默认的坐标系统，可使用 xlim, ylim 等参数，实现**缩放局部**
- `coord_flip()`,
  翻转 x 轴和 y 轴的位置，使得原本在 x 轴上的变量显示在 y 轴上，反之亦然。
- `coord_fixed(ratio = y/x)`
  用特定的长宽比例 (aspect ratio) 作图

1. Nonlinear coordinate system

- `coord_trans(x = "identity", y = "identity", ...)`

x 和 y: 分别指定 x 轴和 y 轴的变换类型。默认为 "identity"，表示不进行变换。

limx,limy: 限制 xy 的显示范围

- coord_polar(theta = "x", start = 0, direction = 1, clip = "on")

创建极坐标图，默认为 coord_polar("x")

柱图变饼图

```
base <- ggplot(mtcars, aes(factor(1), fill = factor(cyl))) +
  geom_bar(width = 1) + theme(legend.position = "none") +
  scale_x_discrete(NULL, expand = c(0, 0)) +
  scale_y_continuous(NULL, expand = c(0, 0))

base + coord_polar(theta = "y") ## 变饼图
```

- coord_map()

  World Map

faceting

panel, strip, axis, tick, tick label, axis label...

- facet_grid(rows ~ cols)

  创建一个由行和列构成的网格布局，使得不同的子图（面板）可以基于数据的一个或多个分类变量进行排列。

  - rows: 行变量，用于在垂直方向上分割面板。
  - cols: 列变量，用于在水平方向上分割面板。

- facet_wrap()

  用于指定行数、列数和方向。

  ```
  facet_wrap(
    facets,
    nrow = NULL,
    ncol = NULL,
    scales = "fixed",
    shrink = TRUE,
    labeller = "label_value",
    as.table = TRUE,
    switch = NULL,
    drop = TRUE,
  ```

```
    dir = "h",
    strip.position = "top"
  )

  # Default parameters are listed.
```

layered grammer (图层语法) 的成分

- 图层 (geom_xxx)
- scale (scale_xxx)
- 坐标系统
- faceting (facet_xxx)

如何在一张图中画多个 panel?

key requirements for multi-panel plots

- order / position
- labeling
- layout

combine multiple plots Useful packages:

- `gridExtra`
- `cowplot`
- `grid`
- `lattice`

`cowplot::plot_grid` parameters

```
plot_grid(
  plot1, plot2,
  ...,
  plotlist = NULL,
  align = c("none", "h", "v", "hv"),
  axis = c("none", "l", "r", "t", "b", "lr", "tb", "tblr"),
  nrow = NULL,
  ncol = NULL,
```

```
  rel_widths = 1,
  rel_heights = 1,
  labels = NULL,
  label_size = 14,
  label_fontfamily = NULL,
  label_fontface = "bold",
  label_colour = NULL,
  label_x = 0,
  label_y = 1,
  hjust = -0.5,
  vjust = 1.5,
  scale = 1,
  greedy = TRUE,
  byrow = TRUE,
  cols = NULL,
  rows = NULL
)
```

- `plot1, plot2, ...`: 这些是你想组合在一起的图形对象。
- `nrow` 和 `ncol`: 指定网格的行数和列数。
- `labels`: 用于每个子图的标签。默认是"AUTO"，自动为每个子图创建标签。
- `label_size`: 设置标签的字体大小。
- `...`: 其他参数，例如 `align` 和 `rel_widths`，用于微调图形的布局。

用 draw_plot 调整 graph 的相对大小

```
plot <-
  ggdraw() +
  draw_plot(plot.iris, x=0, y=.5, width=1, height=0.5) +
  draw_plot(sp, 0, 0, 0.5, 0.5) +
  draw_plot(bp, 0.5, 0, 0.5, 0.5) +
  draw_plot_label(
      c("A", "B", "C"), c(0, 0, 0.5), c(1, 0.5, 0.5), size = 15
  );
```

`draw_plot(plot, x = 0, y = 0, width = 1, height = 1)` 详解:

- plot: the plot to place (ggplot2 or a gtable)
- x: The x location of the lower left corner of the plot.
- y: The y location of the lower left corner of the plot.
- width, height: the width and the height of the plot

use `gridExtra::grid.arrange` to arrange multiple graphs

```
grid.arrange(
    plot1, plot2,
    ...,
    nrow = 1,
    ncol = 1,
    top = NULL,
    bottom = NULL,
    ...)
```

- `plot1`, `plot2`, ... : 这些是你想组合在一起的 grid 图形对象。
- `nrow` 和 `ncol`: 指定网格的行数和列数。
- `top` 和 `bottom`: 可以用来添加顶部和底部标题。
- `...`: 其他参数，可以用于微调图形的布局和样式。
- use `layout_matrix` parameter in `grid.arrange`
  - `layout_matrix` 参数接受一个矩阵，其中的每个元素代表页面上的一个单元格。
  - 你可以通过在这个矩阵中指定数字来控制哪些图形的位置和占据空间。

```
grid.arrange(bp, dp, vp, sc,
             ncol = 2,  ## two columns
             layout_matrix = cbind(c(1,1,1), c(2,3,4)) ## specify the layout
             );
```

Different layouts

Nothing to explain, for it's too intricate.

在图中加入公式和统计信息

Similar to LaTeX

You can just type the formula as exactly what you type in LaTeX, and using `annotiate()` function to add it in your plot.

Remember to attach the library `latex2exp`

```
fig1 =
  fig1 +
  annotate(
    "text",
```

```
    x = 25,
    y = 15,
    label = paste0("y = ", eq, "x + (", intercept, ")\n","Shaded areas are confidence interval
    family = "Aptos Serif"
  )

... +
labs(
    x = TeX("Position of $P_2/\\phi$"),
    y = TeX("Light Intensity/$10^{-7}$A"),
    title = TeX("Intensity of Light with Different $\\phi$")
  )
```

Something about `hjust` and `vjust`

```
td = expand.grid(
    hjust=c(0, 0.5, 1),
    vjust=c(0, 0.5, 1),
    angle=c(0, 45, 90),
    text="text"
)

ggplot(td, aes(x=hjust, y=vjust)) +
    geom_point() +
    geom_text(aes(label=text, angle=angle, hjust=hjust, vjust=vjust)) +
    facet_grid(~angle) +
    scale_x_continuous(breaks=c(0, 0.5, 1), expand=c(0, 0.2)) +
    scale_y_continuous(breaks=c(0, 0.5, 1), expand=c(0, 0.2))
```

In talk09

```
## 计算 ...
m = lm(Fertility ~ Education, swiss);
c = cor.test( swiss$Fertility, swiss$Education );

## 生成公式
eq <- substitute( atop( paste( italic(y), " = ",  a + b %.% italic(x), sep = ""),
                        paste( italic(r)^2, " = ", r2, ", ", italic(p)==pvalue, sep = "" ) ),
                  list(a = as.vector( format(coef(m)[1], digits = 2) ),
                       b =  as.vector(  format(coef(m)[2], digits = 2) ),
                       r2 =  as.vector( format(summary(m)$r.squared, digits = 2) ),
                       pvalue =  as.vector( format( c$p.value , digits = 2) ) ) )
```

```
    );

## 用 as.expression 对公式进行转化  !!!!
eq <- as.character(as.expression(eq));

## 作图，三个图层；特别是 geom_text 使用自己的 data 和 aes ...
ggplot(swiss, aes( x = Education,  y = Fertility ) ) +
        geom_point( shape = 20 ) +
        geom_smooth( se = T ) + ## smooth line ...
        geom_text( data = NULL,
                   aes( x = 30, y = 80, label= eq, hjust = 0, vjust = 1), ## hjust, vjust ???
                   size = 4, parse = TRUE, inherit.aes=FALSE); ## 注意:  parse = TRUE !!!
```

equation 的其它写法（更复杂难懂）

```
## 计算 ...
m = lm(Fertility ~ Education, swiss);
c = cor.test( swiss$Fertility, swiss$Education );

## 生成公式
eq <- substitute( atop( italic(y) == a + b %.% italic(x),
                            italic(r)^2~"="~r2*","~italic(p)==pvalue ),
                        list(a = as.vector( format(coef(m)[1], digits = 2) ),
                             b =  as.vector(  format(coef(m)[2], digits = 2) ),
                             r2 =  as.vector( format(summary(m)$r.squared, digits = 2) ),
                             pvalue =  as.vector( format( c$p.value , digits = 2) ) )
    );
## 用 as.expression 对公式进行转化  !!!!
eq <- as.character(as.expression(eq));

## 作图，三个图层；特别是 geom_text 使用自己的 data 和 aes ...
ggplot(swiss, aes( x = Education,  y = Fertility ) ) +
        geom_point( shape = 20 ) +
        geom_smooth( se = T ) + ## smooth line ...
        geom_text( data = NULL,
                   aes( x = 30, y = 80, label= eq, hjust = 0, vjust = 1), ## hjust, vjust ???
                   size = 4, parse = TRUE, inherit.aes=FALSE); ## 注意:  parse = TRUE !!!
```

分类	R的表达式	显示结果
代数符号	`expression(x + y)`	$x + y$
	`expression(x - y)`	$x - y$
	`expression(x * y)`	xy
	`expression(x / y)`	x/y
	`expression(x %+-% y)`	$x \pm y$
	`expression(x %/% y)`	$x \nabla \cdot y$
	`expression(x %*% y)`	$x \times y$
	`expression(x %.% y)`	$x \cdot y$
	`expression(x[i])`	x_i
	`expression(x^2)`	x^2
	`expression(sqrt(x))`	$\sqrt{x}$
	`expression(sqrt(x,y))`	$\sqrt[y]{x}$
	`expression(list(x,yz))`	x, yz

希腊字符

```
library(ggplot2);
greeks <- c("Alpha", "Beta", "Gamma", "Delta", "Epsilon", "Zeta",
            "Eta", "Theta", "Iota", "Kappa", "Lambda", "Mu",
            "Nu", "Xi", "Omicron", "Pi", "Rho", "Sigma",
            "Tau", "Upsilon", "Phi", "Chi", "Psi", "Omega");

dat <- data.frame( x = rep( 1:6, 4 ), y = rep( 4:1, each = 6), greek = greeks );

plot2 <-
  ggplot( dat, aes(x=x,y=y) ) + geom_point(size = 0) +
    # 画希腊字符，注意下面两行代码的区别
    geom_text( aes( x, y + 0.1, label = tolower( greek ) ), size = 10, parse = T ) +
    geom_text( aes( x, y - 0.1, label = tolower( greek ) ), size = 5  );

##
```

R for bioinformatics, data summarisation and statistics

Talk 10

TOC

- Data summarisation functions (vector data)
  - median, mean, sd, quantile, summary
- Graphical data summarisation (two-D data/ tibble/ table)
  - dot plot
  - smooth
  - linear regression
  - correlation & variance explained
  - groupping & bar/ box/ plots
- statistics
  - parametric tests
    - t-test
    - one way ANNOVA
    - two way ANNOVA
    - linear regression
    - model / prediction / coefficients
  - non-parametric comparison

Vector Summarization

Describe Normal Distribution

You can use `mean` and `sd` to describe normal distributions.

- 是对称的。
- 均值和中位数是一样的。
- 最常见的值接近平均值; 不太常见的价值观与之相去甚远。
- 标准差表示平均值到拐点的距离

Functions to generate random normal distrubions

```
# 生成 10000 个随机数字，使其 mean = 0, sd = 1，且为 normal distribution ...
x <- rnorm(10000, mean = 0, sd = 1);
ggplot( data.frame( data = x ), aes( data ) ) + geom_density(  );
```

Other regular distributions

1. Uniform Distributions 均匀分布

```
# 为向量 q 中的值生成 CDF 概率
pnorm(q, mean = 0, sd = 1)

# 生成向量 p 中概率的分位数
qnorm(p, mean = 0, sd = 1)

# 生成向量 x 中值的概率密度函数
dnorm(x, mean = 0, sd = 1)
```

1. Non-parametric Distributions 非参数分布

```
bi =
        c(7, 3, 2, 1, 7,
    3, 4, 5, 7, 6,
    2, 2, 1, 3, 7,
    2, 6, 8, 2, 7,
    2, 2, 1, 3, 5,
    8, 2, 6, 7, 8,
    6, 2, 8, 7, 9,
    2, 7, 5, 1, 8,
    8, 2, 3, 7, 3
   )
ggplot(
  data.frame(dat = bi),
  aes(dat)) +
geom_density()
```

uniform distribution 的各种函数

注：以下函数中的 n 需要自行决定

```
# generate n random numbers between 0 and 25
runif(n, min = 0, max = 25)
```

```r
# generate n random numbers between 0 and 25 (with replacement)
sample(0:25, n, replace = TRUE)

# generate n random numbers between 0 and 25 (without replacement)
sample(0:25, n, replace = FALSE)
```

other distributions, cont.

```r
n <- 10000;
uni <- tibble( dat = runif(n), type = "uni" );
norm <- tibble( dat = rnorm(n), type = "norm" );
binom <- tibble( dat = rbinom(n, size = 100, prob = 0.5), type = "binom" );
poisson <- tibble( dat = rpois(n, lambda = 4), type = "poisson" );
exp <- tibble( dat = rexp(n, rate = 1) , type = "exp");
gamma <- tibble( dat = rgamma(n, shape = 1) , type = "gamma");
```

量化描述数据

- **mean**: aka average, is the sum of all of the numbers in the data set divided by the size of the data set;
- **median**: The median is the value that is in the middle when the numbers in a data set are sorted in increasing order;
- **sd**: standard deviation;
- **var**: measures how far a set of numbers are spread out;
- **range**: range of values.

量化描述函数

```r
mean( norm$dat );
median( norm$dat );
  mode( norm$dat ); ## 确定给定对象的存储模式 "numeric", "character", "list"

sd(norm$dat);
var(norm$dat);
range(norm$dat);
```

quantile and summary

```r
quantile( norm$dat );
```

```r
## quantile 还接受其它参数
quantile( norm$dat, probs = seq(0, 1, length = 11));

## summary ...
summary( norm$dat );
## summary 也可应用于非数值
summary( combined$type );
## summary 可应用于整个表格; 相当于对每列进行 summary ...
summary( combined );
```

table 函数

返回 vector 当中 unique 值和它们的出现次数

count in dplyr

ntile 函数的参数

... `*tile` 函数都是 equal size

cut 函数

按指定的间隔 (breaks) 对数据进行分割。

不仅可用于 equal distance，还可以用于任意间距

Statistics

Parametric tests

parametric test 的要求

1. 随机取样
2. 值或 residuals 为正态分布；residules 是指观察值与预测值 (mean) 之差
3. 有相同的 variance 方差

how to detect outlier ??

一个很模糊的定义：Outliers are extreme values that fall a long way outside of the other observations. For example, in a normal distribution, outliers may be values on the tails of the distribution.

对于 normal distribution，通常 mean +- 2 or 3 * sd

IQR = s["3rd Qu."]-s["1st Qu."]

outlier: s["1st Qu."] - 1.5 * IQR, s["3rd Qu."] + 1.5 * IQR

`t-test` 检测分布是否与预期一致; e.g., whether the number of steps per day for boys is significantly different from 10,000.

检验评估两个样本的均值是否有显著差异，假设样本服从正态分布，方差近似相等。

There are different types of t-tests in R, depending on the nature of the comparison:

1. **One-Sample t-test:**

   用于确定单个样本的均值与已知或假设的总体均值是否存在显著差异。

   Example:

   ```
   RCopy code
   # One-sample t-test example
   sample_data = c(17, 21, 19, 23, 20, 18, 22)
   t.test(sample_data, mu = 20)
   ```

   This code performs a one-sample t-test on `sample_data` to test if its mean differs significantly from 20.

2. **Independent Samples t-test (or Two-Sample t-test):**

   比较两个独立组的均值，以确定它们是否存在显著差异.

   Example:

   ```
   RCopy code
   # Independent samples t-test example
   group1 = c(23, 25, 28, 22, 20)
   group2 = c(18, 21, 24, 19, 17)
   t.test(group1, group2)
   ```

   This code performs an independent samples t-test on `group1` and `group2` to test if their means are significantly different.

3. **Paired t-test:**

   比较两个相关组的均值（例如，测量前和测量后），以确定它们是否有显著差异。

   Example:

   ```
   RCopy code
   # Paired t-test example
   before = c(32, 28, 30, 29, 31)
   after = c(30, 25, 28, 27, 29)
   t.test(before, after, paired = TRUE)
   ```

This code performs a paired t-test on `before` and `after` to test if there's a significant difference.

The `t.test()` function in R is used to conduct these t-tests. 它返回检验统计量 (t 值)、自由度、p 值和置信区间，从而深入了解观察到的差异是否具有统计显著性。

在 R 中执行 t 检验时，确保满足正态性和等方差的假设对于获得可靠的结果是很重要的。如果违反了这些假设，那么替代检验或数据转换可能更合适。

One-way ANNOVA In R, the one-way analysis of variance (ANOVA) iS 用于检验三个或三个以上独立（不相关）组的均数之间的显著差异。它评估这些群体的平均水平是否彼此显著不同。

The one-way ANOVA assumes that the data meet certain assumptions, including:

- -正态性：每组应遵循正态分布。
- -方差同质性：每组内的方差应大致相等。
- -独立性：每组内的观测值应相互独立。

Here's an example of performing a one-way ANOVA in R:

```r
# Example of one-way ANOVA
group1 = c(15, 20, 25, 30, 35)
group2 = c(10, 18, 25, 32, 40)
group3 = c(12, 22, 28, 32, 38)

# Combining data into a data frame
my_data = data.frame(
  Values = c(group1, group2, group3),
  Group = factor(rep(1:3, each = 5))  # Creating a factor for groups
)

# Performing one-way ANOVA
result_anova = aov(Values ~ Group, data = my_data)
summary(result_anova)
```

Explanation of the code:

1. The data for three groups (`group1`, `group2`, `group3`) are created.
2. The data are combined into a data frame (`my_data`) where the `Values` column contains the measurements and the `Group` column represents the group labels as a factor.

3. The `aov()` function is used to perform the one-way ANOVA, specifying the formula `Values ~ Group`, indicating that `Values` is the dependent variable and `Group` is the independent variable.

4. `summary(result_anova)` provides the ANOVA table with the F-statistic, degrees of freedom, p-value, and other relevant statistics.

The output from `summary(result_anova)` will includeF-统计量、自由度、p-值和组内变异性，允许您确定组间均值是否有显著差异。

若 p 值小于选定的显著性水平（通常为 0.05），则表明组均值之间存在显著差异。Additionally, post-hoc tests like Tukey's HSD test or pairwise t-tests can be performed to identify which specific groups differ significantly from each other after obtaining a significant result in the ANOVA.

Two-way ANNOVA A two-way analysis of variance (ANOVA) in R 用于检验两个分类自变量（因子）之间对连续因变量的交互作用效应。

Here's an example:

```r
# Example of two-way ANOVA
# Assume we have a dataset with 'Treatment', 'Gender', and 'Response' variables

# Creating sample data
set.seed(123)
Treatment = rep(c("A", "B", "C"), each = 20)
Gender = rep(c("Male", "Female"), times = 30)
Response = rnorm(60, mean = c(50, 60, 70), sd = 10)

# Combining data into a data frame
my_data = data.frame(Treatment, Gender, Response)

# Performing two-way ANOVA
result_anova = aov(Response ~ Treatment + Gender + Treatment:Gender, data = my_data)
summary(result_anova)
```

Explanation of the code:

1. Sample data is created with three variables: `Treatment`, `Gender`, and `Response`.

2. The data are combined into a data frame (`my_data`), where `Treatment` and `Gender` are categorical factors, and `Response` is the continuous dependent variable.

3. The `aov()` function performs the two-way ANOVA. The formula `Response ~ Treatment + Gender + Treatment:Gender` specifies the main effects of `Treatment` and `Gender`, as well as their interaction effect.

4. `summary(result_anova)` provides the ANOVA table with F-statistics, degrees of freedom, p-values, and other statistics for each factor and their interaction.

The output from `summary(result_anova)` will include 治疗和性别的主要影响的信息，以及它们之间的交互作用效果。它支持您确定每个因子是否独立存在显著效应，以及它们的交互作用是否显著影响" 响应" 变量。

The interpretation of a two-way ANOVA 包括分析与每个因子及其交互作用相关联的 p 值。显著的 p 值表示相应的因子或交互作用对因变量有显著的影响。

Additionally, post-hoc tests or further analyses can be conducted to explore specific comparisons between groups or factors after obtaining significant results in the ANOVA.

Linear Regression 线性回归是一种统计方法，通过对观测数据拟合线性方程来模拟因变量和一个或多个自变量之间的关系。

In R, linear regression can be performed using the `lm()` function, which stands for "linear model."

Here's an example:

```r
# Example of simple linear regression
# Suppose we have a dataset with 'x' as the independent variable and 'y' as the dependent vari

# Creating sample data
set.seed(123)
x = 1:50
y = 2 * x + rnorm(50, mean = 0, sd = 5)  # Generating 'y' as a linear function of 'x' with som

# Creating a data frame
my_data = data.frame(x, y)

# Performing linear regression
model = lm(y ~ x, data = my_data)
summary(model)
```

Explanation of the code:

1. Sample data is generated with an independent variable `x` and a dependent variable `y`. In this example, `y` is generated as a linear function of `x` with some added noise using `rnorm()` to simulate real-world variability.
2. The data are combined into a data frame `my_data`.
3. The `lm()` function fits a linear regression model where `y` is the dependent variable and `x` is the independent variable (`y ~ x`). The argument `data = my_data` specifies the data frame containing the variables.

4. `summary(model)` provides a summary of the linear regression model, including coefficients, standard errors, t-values, p-values, R-squared, and other statistics.

Interpreting the output from `summary(model)`:

- The coefficients section shows the estimated coefficients for the intercept and the slope of the regression line (`Intercept` and `x`).
- The p-values associated with the coefficients indicate the significance of each variable in predicting the dependent variable. Lower p-values suggest stronger evidence against the null hypothesis of no effect.
- The R-squared value represents the proportion of variance in the dependent variable explained by the independent variable(s). Higher R-squared values indicate a better fit of the model to the data.

Linear regression in R can also 通过在模型中包含多个自变量，将其扩展到多元线性回归 (`y ~ x1 + x2 + ...`).

Additionally, diagnostic plots and further analyses can be performed to assess model assumptions and goodness of fit.

Model / Prediction / Coefficients In linear regression, the model equation is expressed as:

$$[y = \beta_0 + \beta_1 \cdot x_1 + \beta_2 \cdot x_2 + ... + \beta_n \cdot x_n + \epsilon]$$

Where:

- (y) is the dependent variable.
- ($x_1, x_2, ..., x_n$) are the independent variables.
- (β_0) is the intercept (constant term).
- ($\beta_1, \beta_2, ..., \beta_n$) are the coefficients (slope parameters) that represent the change in (y) associated with a one-unit change in the corresponding (x) variable, assuming all other variables remain constant.
- (ϵ) represents the error term.

In R, after fitting a linear regression model using `lm()`:

```
# Assuming 'model' is the linear regression model obtained previously
summary(model)
```

The output from `summary(model)` provides information including:

- **Coefficients:** 本节显示每个自变量的估计系数（" 估计"），包括截距。这些系数代表因变量的估计变化，对应的自变量的一个单位的变化，保持其他变量不变。
- **Residuals:** 残差代表因变量的观测值和预测值之间的差异。这些用于评估模型的拟合优度。
- **R-squared:** 指示由自变量解释的因变量中的方差比例。较高的值指示模型对数据的拟合更好。

After obtaining the coefficients from the model, predictions can be made using new or existing data:

```
# Assuming 'new_data' contains the new data for prediction
predicted_values = predict(model, newdata = new_data)
```

Replace `new_data` with the data for which you want to make predictions. The `predict()` function uses the coefficients from the model to generate predicted values of the dependent variable based on the independent variables in the new data.

The coefficients obtained from the linear regression model (`model$coefficients`) represent the slopes of the regression line and the intercept, which are crucial for predicting new values and understanding the relationships between variables in the model.

Non-parametric Comparison

非参数方法是当参数方法所要求的正态性、方差齐性或线性假设被数据违反或不满足时所使用的统计技术。这些方法不依赖于特定的总体分布假设，对于分析可能不遵循正态分布的数据是有用的。

Here are some commonly used non-parametric methods for comparison in R:

1. **Mann-Whitney U Test (Wilcoxon Rank-Sum Test):**

   用于在 t 检验的假设不满足时比较两个独立的组。

   Example:

   ```
   # Assuming 'group1' and 'group2' are vectors of numeric data
   wilcox.test(group1, group2)
   ```

2. **Kruskal-Wallis Test:**

   Mann-Whitney U 检验的扩展，用于比较两个以上的独立组。

   Example:

   ```
   # Assuming 'group1', 'group2', 'group3' are vectors of numeric data
   kruskal.test(list(group1, group2, group3))
   ```

3. **Wilcoxon Signed-Rank Test:**

   用于比较配对样本或相关样本。

   Example:

   ```r
   # Assuming 'before' and 'after' are vectors of paired numeric data
   wilcox.test(before, after, paired = TRUE)
   ```

4. **Mood's Median Test:**

   测试两个或多个独立组中的中位数是否相等。

   Example:

   ```r
   # Assuming 'group1', 'group2', 'group3' are vectors of numeric data
   median_test = mood.test(group1, group2, group3)
   median_test
   ```

5. **Friedman Test:**

   Wilcoxon Signed-Rank 检验的扩展，用于比较两个以上配对或相关组。

   Example:

   ```r
   # Assuming 'group1', 'group2', 'group3' are matrices or data frames of paired numeric dat
   friedman.test(group1, group2, group3)
   ```

这些非参数检验提供了传统的参数检验的替代品，并对违反某些假设是强大的。它们在处理有序或倾斜数据或样本容量较小时特别有用，因为它们比参数检验依赖更少的分布假设。

Linear and nonlinear regression

Talk 11

This topic is particularly complex, so it is voluminous and obscure.

TOC

- linear regression
- nonlinear regression
- modeling and prediction
- **K-fold** & **X times** cross-validation
- external validation

Linear Regression

What is linear regression?

线性回归是一种利用数理统计中的回归分析来确定两个或多个变量之间相互依赖的数量关系的统计分析方法。

- `Y` can be explained by a variable `X`: One-way Linear Regression
- `Y` can be explained by multiple variables such as `X`, `Z`: Multiple Linear Regression

Linear regression in R is typically performed using the `lm()` function, which stands for "linear model." Here's how to fit a linear regression model in R and some useful functions related to linear regression:

Fitting a Linear Regression Model:

```r
# Example of fitting a linear regression model
# Assuming 'my_data' is a data frame with 'x' as the independent variable and 'y' as the depen
model = lm(y ~ x, data = my_data)
summary(model)  # Display summary statistics of the model
```

- **`lm()` Function:** Fits a linear regression model. The formula `y ~ x` specifies that `y` is the dependent variable and `x` is the independent variable. `data = my_data` indicates the data frame containing the variables.
- **`summary()` Function:** Displays a summary of the linear regression model, including coefficients, standard errors, t-values, p-values, R-squared, and other statistics.

glm vs. lm

```
lm(formula, data, ...)
glm(formula, family=gaussian, data, ...)
```

glm:

1. 当 `family=gaussian` 时，二者是一样的。

```
library(texreg);
m.lm <- lm(am ~ disp + hp, data=mtcars);
m.glm <- glm(am ~ disp + hp, data=mtcars);
screenreg(l = list(m.lm, m.glm))
```

glm 还可用于其它类型数据的分析

1. Logistic regression (family=binomial)

预测的结果（Y）是 binary 的分类，比如 `Yes`, `No`，且只能有两个值；

```
dat <- iris %>% filter( Species %in% c("setosa", "virginica") );
bm <- glm( Species ~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width,
           data = dat, family = binomial );

data.frame( predicted = bm %>% predict( dat, type = "response" ),
            original = dat$Species ) %>% sample_n(6) %>% arrange( original );
```

注意:

`predict(., type = "response")` 指示 `predict()` 函数返回模型的响应值。

对于逻辑回归模型，这意味着返回的是概率而不是对数几率。

glm 的 Poisson regression (family=poisson)

泊松回归是一种特殊类型的回归，其中响应变量由**计数数据组成**.

Asumptions:

1. 响应变量由计数数据组成。
2. 观察是独立的。
3. 模型的均值和方差相等。
4. 计数的分布遵循泊松分布。

Other Useful Functions for Linear Regression Analysis:

1. **`coefficients()` Function:**

   检索线性回归模型的系数。

   Example:

   ```
   coef = coefficients(model)
   coef
   ```

2. **`predict()` Function:**

   使用拟合模型生成预测。

   Example:

   ```
   new_data = data.frame(x = c(10, 20, 30))  # New data for prediction
   predicted_values = predict(model, newdata = new_data)
   predicted_values
   ```

3. **`residuals()` Function:**

   检索残差（观测值和预测值之间的差异）。

   Example:

   ```
   residuals = residuals(model)
   residuals
   ```

4. **`fitted()` Function:**

   检索拟合（预测）值。

   Example:

   ```
   fitted_values = fitted(model)
   fitted_values
   ```

5. **`vcov()` Function:**

   计算系数的方差-协方差矩阵。

   Example:

   ```
   var_cov_matrix = vcov(model)
   var_cov_matrix
   ```

6. **`anova()` Function:**

   对拟合模型执行方差分析（ANOVA）。

   Example:

   ```
   anova_table = anova(model)
   anova_table
   ```

7. **`summary()` Function:**

   提供拟合模型的概述，包括参数估计值、标准误差、收敛信息和拟合优度统计量。

   Example:

   ```
   summary(model)
   ```

8. **`coef()` Function:**

   从拟合模型中提取估计系数。

   Example:

   ```
   coef(model)
   ```

9. **`confint()` Function (for prediction intervals):**

   计算拟合模型中预测值的预测区间。

   Example:

   ```
   prediction_intervals = predict(model, interval = "prediction")
   prediction_intervals
   ```

10. **`deviance()` Function:**

    计算拟合模型的偏差，这是缺乏拟合的度量。

    Example:

    ```
    deviance(model)
    ```

11. **`update()` Function:**

    允许使用不同的设置或数据更新或改装模型。

    Example:

    ```
    updated_model = update(model, start = list(a = 2, b = 2, c = 2))
    summary(updated_model)
    ```

12. **`AIC()` and `BIC()` Functions:**

    计算 Akaike 信息准则（阿灵顿资产投资）和贝叶斯信息准则（BIC）来评估模型质量和比较模型。

    Example:

    ```
    AIC(model)
    BIC(model)
    ```

13. **`plot()` Function (for diagnostic plots):**

    生成诊断图以评估模型的充分性（例如，残差与拟合值、Q-Q 图）。

    Example:

```
plot(model)
```

These functions help in obtaining and analyzing various aspects of the linear regression model, such as coefficients, predictions, residuals, variance-covariance matrix, and ANOVA tables, aiding in model interpretation, examining the fitted model's statistics, coefficients, goodness-of-fit measures, prediction intervals, model comparisons, and diagnostics, allowing for a comprehensive analysis of nonlinear regression models in R.

Nolinear Regression

当变量之间的关系不能用线性模型充分描述时，使用非线性回归。在 R 中，拟合非线性模型包括估计参数来描述变量之间的非线性关系。Here's an example using the `nls()` function, along with some relevant functions for nonlinear regression analysis:

Fitting a Nonlinear Regression Model:

1. `nls( equation, data = data, start = ... )`
2. `y~a*x/(b+x)`

```
# Example of fitting a nonlinear regression model (assuming a quadratic function)
# Assuming 'my_data' is a data frame with 'x' as the independent variable and 'y' as the depen

# Fitting a quadratic model: y = a * x^2 + b * x + c
model = nls(y ~ a * I(x^2) + b * x + c, data = my_data, start = list(a = 1, b = 1, c = 1))
summary(model)  # Display summary statistics of the model
```

- **`nls()` Function:** Fits a nonlinear regression model. The formula `y ~ a * I(x^2) + b * x + c` specifies a quadratic function. `data = my_data` indicates the data frame containing the variables, and `start = list(a = 1, b = 1, c = 1)` provides initial parameter values.
- **`summary()` Function:** Displays a summary of the nonlinear regression model, including parameter estimates, standard errors, t-values, convergence information, and other statistics.

Other Useful Functions for Nonlinear Regression Analysis:

1. **`predict()` Function:**

   Generates predictions using the fitted nonlinear model.

   Example:

```r
new_data = data.frame(x = c(10, 20, 30))  # New data for prediction
predicted_values = predict(model, newdata = new_data)
predicted_values
```

2. **`residuals()` Function:**

   Retrieves the residuals (differences between observed and predicted values).

   Example:

```r
residuals = residuals(model)
residuals
```

3. **`confint()` Function:**

   Computes confidence intervals for model parameters.

   Example:

```r
conf_intervals = confint(model)
conf_intervals
```

4. **`nls.control()` Function:**

   为 `nls()` 函数提供控制参数，允许对非线性拟合过程进行调整。

   Example:

```r
control_params = nls.control(maxiter = 100, tol = 1e-6)
model = nls(y ~ a * I(x^2) + b * x + c, data = my_data, start = list(a = 1, b = 1, c = 1)
```

5. **`anova()` Function:**

   Performs analysis of variance (ANOVA) for the fitted nonlinear model.

   Example:

```r
anova_table = anova(model)
anova_table
```

6. **`nlsLM()` Function (from the `minpack.lm` package):**

   - An alternative to `nls()` that provides enhanced convergence properties and extended functionality for nonlinear least squares.

   Example:

```r
library(minpack.lm)
model_nlsLM = nlsLM(y ~ a * I(x^2) + b * x + c, data = my_data, start = list(a = 1, b = 1
summary(model_nlsLM)
```

7. **`augment()` Function (from the `broom` package):**

   - 使用其他列（如拟合/预测值、残差和其他模型信息）创建整洁的数据框架。

Example:

```
library(broom)
augmented_model = augment(model)
head(augmented_model)
```

8. **`glance()` Function (from the `broom` package):**
   - 提取模型级统计信息，以简洁的格式提供模型的摘要。

   Example:

   ```
   glance_summary = glance(model)
   glance_summary
   ```

9. **`tidy()` Function (from the `broom` package):**
   - 将模型系数和相关统计数据提取到一个整洁的数据框架中。

   Example:

   ```
   tidy_summary = tidy(model)
   tidy_summary
   ```

10. **`nlstools::nlstools()` Function:**
    - 为非线性回归模型提供诊断工具和可视化，帮助模型评估。

    Example:

    ```
    library(nlstools)
    model_tools = nlstools(model)
    plot(model_tools)
    ```

11. **`nls2()` Function:**
    - 提供具有多个起始值的'nls（）'的扩展版本，以改进收敛性。

    Example:

    ```
    model_nls2 = nls2(y ~ a * I(x^2) + b * x + c, data = my_data, start = list(a = 1, b = 1,
    summary(model_nls2)
    ```

These functions are commonly used for nonlinear regression analysis in R, helping in prediction, residual analysis, confidence interval computation, and controlling the fitting process.

Modeling and Prediction

When it comes to modeling and prediction using regression analysis, especially nonlinear regression, understanding the model, making predictions, and assessing the model's performance are crucial. Here's a step-by-step guide on how to approach modeling and prediction:

Modeling and Prediction Steps:

1. **Data Preparation:**

Prepare your dataset with variables for the dependent (response) and independent (predictor) variables.

1. **Fitting the Nonlinear Model:**

使用 `nls()` 或类似函数拟合非线性回归模型，指定适当的公式和初始参数值。

Example:

```
model =
nls(y ~ a * I(x^2) + b * x + c,
    data = my_data,
    start = list(a = 1, b = 1, c = 1))
```

1. **Model Summary and Assessment:**

Use `summary()` and other functions (`glance()`, `tidy()`, etc.) 以获得模型的概述，包括系数、拟合优度度量和诊断。

Example:

```
summary(model)
```

1. **Prediction with the Fitted Model:**

使用拟合模型为新数据或现有数据生成预测。

Example:

```
new_data = data.frame(x = c(10, 20, 30))  # New data for prediction
predicted_values = predict(model, newdata = new_data)
predicted_values
```

1. **Model Evaluation:**

Evaluate the model's performance using various metrics (e.g., residuals, R-squared, RMSE) and diagnostic plots (e.g., residuals vs. fitted values, Q-Q plots) to assess how well the model fits the data.

1. **Adjustment and Refinement:**

Depending on the evaluation results, consider refining the model by adjusting parameters, exploring different models, or including/excluding variables to improve performance.

1. **Prediction Intervals and Uncertainty:**

Compute prediction intervals using `predict()` with `interval = "prediction"` to quantify the uncertainty around predictions.

1. **Model Comparison (if applicable):**

Compare multiple models using metrics like AIC, BIC, or likelihood ratio tests to select the most appropriate model.

By following these steps, you'll be able to build, evaluate, and utilize nonlinear regression models for prediction in R effectively. Remember, interpreting the model's results, assessing its assumptions, and validating predictions are crucial aspects of regression analysis.

K-fold & X times cross-validation

K-fold 交叉验证和 X 次交叉验证都是用于评估机器学习模型（包括回归模型）的性能和泛化能力的技术，方法是将数据划分为子集进行训练和验证。

K-fold Cross-Validation:

k-fold 交叉验证涉及将数据集拆分成 K 个大小相等的折叠。对模型进行 K 次训练，每次使用 K-1 折叠进行训练，其余折叠进行验证。这个过程确保每个数据点都用于训练和验证。

Steps for K-fold Cross-Validation:

1. **Partition Data:** 将数据集分割成 K 个大小相等的折叠。
2. **Model Training:** 对模型进行 K 次训练，每次使用 K-1 折叠作为训练数据。
3. **Validation:** 验证模型在其余部分（训练中未使用）上的性能，并计算评估指标。.
4. **Average Metrics:** 对 K 个迭代中的评估指标求平均值，以获得对模型性能的总体评估。

X times Cross-Validation:

X 次交叉验证，也被称为重复的 K-fold 交叉验证，类似于 K-fold 的交叉验证，但这个过程是重复 X 次。它重复创建随机分区的数据到 K 折叠，训练模型，并评估其性能。该方法通过对多次试验的结果求平均值来提供更稳健的模型性能估计。

Steps for X times Cross-Validation:

1. **Partition Data Repeatedly:** 将数据集随机分成 K 个折叠，X 次。
2. **Model Training:** 为每个迭代训练模型，使用 K-1 折叠进行训练。
3. **Validation:** 验证模型在剩余折叠上的性能并计算评估指标。
4. **Average Metrics:** 对 X 次迭代中的评估指标求平均值，以获得更稳定、更可靠的模型性能估计值。

Implementation in R:

In R, you can perform K-fold and X times cross-validation using functions from various packages like `caret`, `rsample`, or `crossval`. For example, using `caret`:

K-fold Cross-Validation in R with `caret`:

```r
library(caret)
# Define a train control using k-fold cross-validation
train_control =
        trainControl(method = "cv", number = K)
# Specify K for the number of folds

# Train the model using k-fold cross-validation
model =
        train(
    formula,
    data = my_data,
    method = "lm",
    trControl = train_control
  )
```

X times Cross-Validation in R with `caret`:

```r
library(caret)
# Define a train control using repeated k-fold cross-validation
train_control =
trainControl(
  method = "repeatedcv",
  number = K, repeats = X)
# Specify K for folds and X for repeats

# Train the model using repeated k-fold cross-validation
```

```
model =
        train(
    formula,
    data = my_data,
    method = "lm",
    trControl = train_control
  )
```

Replace `formula` and `my_data` with the appropriate regression formula and your dataset. Adjust the parameters `K` and `X` according to your preferences for the number of folds and repetitions.

这些交叉验证技术有助于估计模型的性能，并帮助评估模型如何推广到看不见的数据，从而提供洞察其鲁棒性和可靠性。调整这些技术可以提高回归模型的准确性和稳定性的评价.

External Validation

外部验证是指使用在模型开发过程中未使用的独立数据集来评估预测模型的性能。它是评估一个模型如何从不同的来源或时间段推广到新的、看不见的数据的关键步骤，以验证它在实际应用中的可靠性和健壮性。

Steps for External Validation:

1. **Obtain an Independent Dataset:** 获取与用于模型训练和验证的数据集不同的单独数据集。理想情况下，这个数据集应该代表相同的问题或领域，但来自不同的源、时间框架或人群。
2. **Preprocess Data:** 对独立数据集进行类似于训练数据集的预处理（例如，处理缺失值、编码分类变量、缩放特征），以确保兼容性。
3. **Apply Trained Model:** 使用在原始数据集上训练的模型对独立数据集进行预测。
4. **Evaluate Model Performance:** 使用相关评估指标（例如准确度、RMSE、精度、召回率）评估模型在独立数据集上的性能，并将这些指标与在训练/验证数据集上实现的性能进行比较。
5. **Analyze Results:** 分析从独立数据集获得的性能指标，以确定模型是否保持其预测能力和泛化能力。性能良好的独立数据集上的模型表明，良好的泛化能力和可靠性。

Implementation in R:

In R, the process involves loading the trained model and applying it to the independent dataset for prediction. Use appropriate evaluation functions (`predict()`, evaluation metrics) to assess the model's performance on the external dataset.

Example in R:

```
# Load the trained model (replace 'model' with your trained model)
load("trained_model.RData")

# Load and preprocess the independent dataset
# (replace 'independent_data.csv' with your dataset)
independent_data = read.csv("independent_data.csv")
# Perform similar preprocessing steps as used for the training data

# Apply the trained model to the independent dataset for prediction
predicted_values = predict(model, newdata = independent_data)

# Evaluate model performance on the independent dataset
# Use appropriate evaluation metrics
# (e.g., RMSE, accuracy)
# and compare with training/validation results
```

Replace the file paths, data loading, and evaluation steps with your specific dataset and evaluation procedures. Ensure the compatibility of the independent dataset with the preprocessing steps applied to the original dataset for accurate evaluation.

Machine learning basics

Talk 12

This topic is particularly complex, so it is voluminous and obscure.

TOC

- Machine Learning Algorithms Generalization
- Random Forest
- Feature Selection

机器学习可分为以下几类

\FontSmall

1）**回归算法**

2）基于实例的算法

3）**决策树学习**

4）贝叶斯方法

5）基于核的算法

6）**聚类算法**

7）**降低维度算法**

8）关联规则学习

9）集成算法

10）人工神经网络

1. 回归算法

回归算法是试图采用对误差的衡量来探索变量之间的关系的一类算法。常见的回归算法包括：

- 最小二乘法（Ordinary Least Square），
- 逻辑回归（Logistic Regression），
- 逐步式回归（Stepwise Regression），
- 多元自适应回归样条（Multivariate Adaptive Regression Splines）以及
- 本地散点平滑估计（Locally Estimated Scatterplot Smoothing）。

2. 基于实例的算法

基于实例的算法常常用来对决策问题建立模型，这样的模型常常先选取一批样本数据，然后根据某些近似性把新数据与样本数据进行比较。通过这种方式来寻找最佳的匹配。

因此，基于实例的算法常常也被称为“赢家通吃”学习或者“基于记忆的学习”。常见的算法包括：

- k-Nearest Neighbor(KNN),
- 学习矢量量化（Learning Vector Quantization，LVQ），以及
- 自组织映射算法（Self-Organizing Map，SOM）。

深度学习的概念源于人工神经网络的研究。含多隐层的多层感知器就是一种深度学习结构。深度学习通过组合低层特征形成更表示属性类别或特征，以发现数据的分布式特征表示。

3. 决策树学习

决策树算法根据数据的属性采用树状结构建立决策模型，决策树模型常常用来解决分类和回归问题。常见的算法包括：

- 分类及回归树（Classification And Regression Tree，CART），
- ID3 (Iterative Dichotomiser 3)，
- C4.5，Chi-squared Automatic Interaction Detection(CHAID),
- Decision Stump, 随机森林（Random Forest），
- 多元自适应回归样条（MARS）以及
- 梯度推进机（Gradient Boosting Machine，GBM）。

4. 贝叶斯方法

贝叶斯方法算法是基于贝叶斯定理的一类算法，主要用来解决分类和回归问题。常见算法包括：

- 朴素贝叶斯算法，
- 平均单依赖估计（Averaged One-Dependence Estimators，AODE），以及
- Bayesian Belief Network（BBN）。

5. 基于核的算法

基于核的算法中最著名的莫过于支持向量机（SVM）了。基于核的算法把输入数据映射到一个高阶的向量空间，在这些高阶向量空间里，有些分类或者回归问题能够更容易的解决。常见的基于核的算法包括：

- 支持向量机（Support Vector Machine，SVM），
- 径向基函数（Radial Basis Function，RBF），以及
- 线性判别分析（Linear Discriminate Analysis，LDA）等。

6. 聚类算法

聚类，就像回归一样，有时候人们描述的是一类问题，有时候描述的是一类算法。聚类算法通常按照中心点或者分层的方式对输入数据进行归并。所以的聚类算法都试图找到数据的内在结构，以便按照最大的共同点将数据进行归类。常见的聚类算法包括

- k-Means 算法以及
- 期望最大化算法（Expectation Maximization，EM）。

7. 降低维度算法

像聚类算法一样，降低维度算法试图分析数据的内在结构，不过降低维度算法是以非监督学习的方式试图利用较少的信息来归纳或者解释数据。这类算法可以用于高维数据的可视化或者用来简化数据以便监督式学习使用。常见的算法包括：

- 主成份分析（Principle Component Analysis，PCA），
- 偏最小二乘回归（Partial Least Square Regression，PLS），
- Sammon 映射，
- 多维尺度（Multi-Dimensional Scaling, MDS），
- 投影追踪（Projection Pursuit）等。

8. 关联规则学习

关联规则学习通过寻找最能够解释数据变量之间关系的规则，来找出大多元数据集中有用联规则。常见算法包括

- Apriori 算法和
- Ec t 算法等。

9. 集成算法

集成算法用一些相对较弱的学习模型独立地就同样的样本进行训练，然后把结果整合起来进行整体预测。集成算法的主要难点在于究竟集成哪些独立的较弱的学习模型以及如何把学习结果整合起来。这是一类非常强大的算法，同时也非常流行。常见的算法包括：

- Boosting，
- Bootstrapped Aggregation（Bagging），
- AdaBoost，
- 堆叠泛化（Stacked Generalization，Blending），
- 梯度推进机（Gradient Boosting Machine, GBM），
- 随机森林（Random Forest）

10. 人工神经网络

人工神经网络算法模拟生物神经网络，是一类模式匹配算法。通常用于解决分类和回归问题。人工神经网络是机器学习的一个庞大的分支，有几百种不同的算法。（其中**深度学习**就是其中的一类算法），重要的人工神经网络算法包括：

- 感知器神经网络（Perceptron Neural Network），
- 反向传递（Back Propagation），
- Hopfield 网络，
- 自组织映射（Self-Organizing Map, SOM）。
- 学习矢量量化（Learning Vector Quantization，LVQ）。

section 3: 随机森林

随机森林 – Random forest

本文大部分内容取自 “easyai.tech” 网站的《随机森林 – Random forest》一文，有修改。

随机森林是一种由**决策树**构成的集成算法，适用于**小样本**数据，在很多情况下都能有不错的表现。

随机森林属于 **集成学习**中的 Bagging（Bootstrap AGgregation 的简称）方法。

决策树 - decision tree

决策树是一种很简单的算法，他的解释性强，也符合人类的直观思维。这是一种基于 if-then-else 规则的有监督学习算法，

Steps to Implement Random Forest in R:

1. Load Required Library: Start by loading the necessary library (`randomForest`) if not already installed.

```
install.packages("randomForest")  # Install package if not installed
library(randomForest)
```

2. Prepare Data: 将数据集加载到 R 中并执行必要的预处理步骤，如处理缺失值、对分类变量进行编码以及将数据拆分为训练集和测试集。

```
# Example: Assuming 'my_data' is your dataset
# Split data into features (X) and target variable (Y)
X = my_data[, -target_column_index]  # Features
Y = my_data[, target_column_index]   # Target variable
```

3. Train the Random Forest Model: Use the `randomForest()` function to 通过指定公式和训练数据来训练模型。

```
# Example: Training a Random Forest model for regression
model = randomForest(Y ~ ., data = my_data)
# For classification: model = randomForest(factor(Y) ~ ., data = my_data)
```

4. Model Tuning (Optional): Adjust hyperparameters such as the number of trees (`ntree`), maximum depth (`max_depth`), and others to optimize model performance.

```
# Example: Setting number of trees and maximum depth
model = randomForest(Y ~ ., data = my_data, ntree = 100, max_depth = 10)
```

5. Make Predictions: Use the trained model to make predictions on new or test data.

```
# Example: Making predictions on test data
predicted_values = predict(model, newdata = test_data)
```

6. Model Evaluation: Evaluate the model's performance using appropriate metrics (e.g., RMSE for regression, accuracy, confusion matrix for classification) on test/validation data.

```
# Example: Evaluate model performance (for regression)
error = sqrt(mean((predicted_values - true_values)^2))  # Calculate RMSE
```

Example - Random Forest for Regression:

Here's an example using a built-in dataset (`mtcars`) in R for regression:

```
library(randomForest)

# Load dataset
data(mtcars)

# Split data into features and target variable
X = mtcars[, -1]  # Exclude the first column (target variable)
Y = mtcars[, 1]   # First column (target variable)

# Train the Random Forest model
model = randomForest(Y ~ ., data = mtcars)

# Make predictions on the same dataset (for demonstration)
predicted_values = predict(model, newdata = mtcars)

# Evaluate model performance (RMSE)
error = sqrt(mean((predicted_values - mtcars$mpg)^2))
```

Replace `my_data`, `test_data`, and the respective column names with your specific dataset and target variable. Adjust hyperparameters based on your problem and dataset characteristics for better model performance. Additionally, use appropriate evaluation metrics for assessing model accuracy and performance.

Feature Selection

特征选择是机器学习中的一个关键步骤，旨在选择最相关、最具信息量的特征，以提高模型性能、降低计算复杂度并减轻过拟合。在 R 中，各种方法可以帮助识别重要的特征。

Feature Selection Techniques in R:

1. Correlation Analysis: 计算特征与目标变量之间或特征本身之间的相关系数，以识别高度相关的特征。删除冗余或高度相关的功能。

```
# Example: Using cor() for correlation analysis
correlation_matrix = cor(my_data)
```

2. Filter Methods: 使用统计方法（例如，卡方检验，互信息），以排名和选择的基础上，他们的个人相关的目标变量的功能。

```r
# Example: Using chi-squared test for feature selection
library(caret)
feature_selection = nearZeroVar(my_data)
```

3. Wrapper Methods: 通过迭代训练模型并选择可实现最佳模型性能的子集来评估特征子集。

```r
# Example: Using recursive feature elimination (RFE)
library(caret)
control = rfeControl(functions = rfFuncs, method = "cv", number = 10)
feature_selection = rfe(X, Y, sizes = c(1:10), rfeControl = control)
```

4. Embedded Methods: 模型训练期间的特征选择，其中算法本身执行特征选择（例如，LASSO、随机森林、梯度提升）。

```r
# Example: Using Random Forest for feature selection
library(randomForest)
model = randomForest(Y ~ ., data = my_data)
importance = importance(model)
```

5. Dimensionality Reduction Techniques: 主成分分析（PCA）或 t-Distributed 随机近邻嵌入（t-SNE）等技术通过创建捕获大部分原始信息的新变量来缩小特征空间。

```r
# Example: PCA for feature reduction
pca_result = prcomp(my_data, scale. = TRUE)
```

根据数据集的特征、问题复杂性和计算资源选择特征选择方法。多种技术的组合通常会产生最好的结果，因为不同的方法可以捕获特征重要性的不同方面。进行试验和迭代，为您的机器学习模型找到最适合的功能。

This is the end of the article. Take a rest~