Lab01 : Data manipulation and ggplot

Yalin Yang

2020-02-28

# Task 1: Statistical Graphs (3 pts)

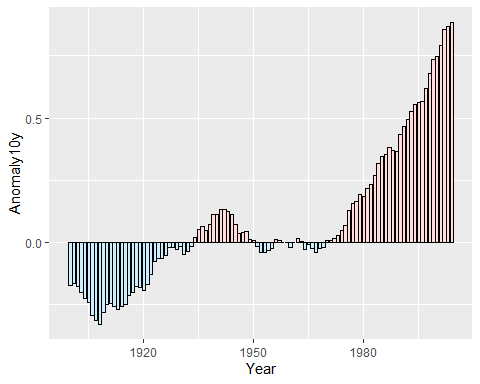
The following graphs are reproductions from the [R Graphics Cookbook, 2e](https://r-graphics.org/index.html). One of the objectives of task 1 is that you visit the **R Graphics Cookbook** and explore the graphs on display there.

Study the sections associated with the graphs which are displayed in task 1.1 to 1.6. Its sections document how these graphs were build. For *educational purposes* you may want to read the associated documentation of the employed functions and experiment with their options.

You job in task 1 is to reproduce these graphs and show the code, which generated them. The data used in the **R Graphics Cookbook** are available in library("gcookbook").

## Task 1.1 (0.5 pts)

climate\_sub <- climate %>% filter(Source == "Berkeley" & Year >= 1900) %>%   
 mutate(pos=Anomaly10y >= 0)  
  
ggplot(climate\_sub, aes(x=Year, y=Anomaly10y, fill=pos)) +  
 geom\_col(position="identity", colour="black", size=0.25) +  
 scale\_fill\_manual(values = c("#CCEEFF","#FFDDDD"), guide=FALSE)



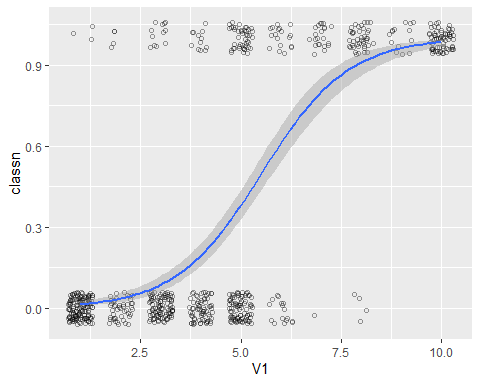
## Task 1.2 (0.5 pts)

library(MASS)

##   
## Attaching package: 'MASS'

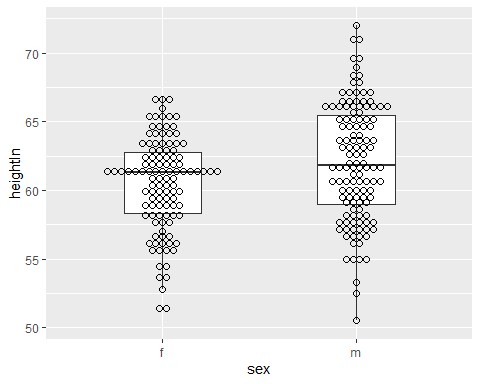
## The following object is masked from 'package:dplyr':  
##   
## select

biopsy\_mod <- biopsy %>% mutate(classn = recode(class, benign=0, malignant=1))  
  
ggplot(biopsy\_mod, aes(x=V1, y=classn))+  
 geom\_point(position=position\_jitter(width=0.3, height=0.06), alpha=0.4, shape=21, size=1.5)+  
 stat\_smooth(method=glm, method.args=list(family=binomial))



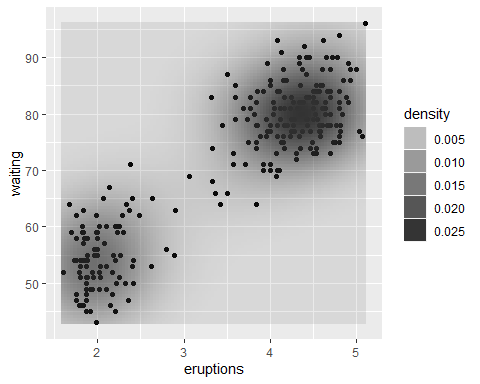
## Task 1.3 (0.5 pts)

ggplot(heightweight, aes(x=sex, y=heightIn)) +  
 geom\_boxplot(outlier.colour = NA, width= 0.4)+  
 geom\_dotplot(binaxis = "y", binwidth = 0.5, stackdir = "center", fill=NA)



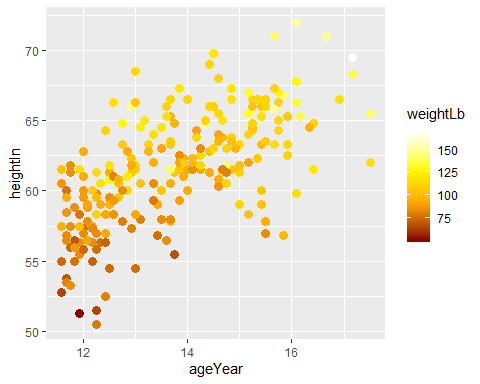
## Task 1.4 (0.5 pts)

ggplot(faithful, aes(x=eruptions, y=waiting))+  
 geom\_point()+  
 stat\_density2d(aes(alpha=..density..), geom="tile", contour = FALSE)



## Task 1.5 (0.5 pts)

library(scales)  
ggplot(heightweight, aes(x=ageYear, y=heightIn, colour=weightLb))+  
 geom\_point(size=3)+  
 scale\_colour\_gradientn(colours = c("darkred","orange","yellow","white"))



## Task 1.6 (0.5 pts)

For this map you would need to setup the data with

library(maps)  
library(mapproj)  
states\_map <- map\_data("state")  
crimes <- data.frame(state=tolower(rownames(USArrests)),USArrests)  
crime\_map <- merge(states\_map, crimes, by.x="region", by.y="state")  
  
qa <- quantile(crimes$Assault, seq(0,1,by=0.2))  
crimes$Assault\_q <- cut(crimes$Assault, qa, labels=c("0-20%","20-40%","40-60%","60-80%","80-100%"),  
 include.lowest = TRUE)  
pal <- colorRampPalette(c("#559999","grey80","#BB650B"))(5)  
ggplot(crimes, aes(map\_id=state, fill=Assault\_q))+  
 geom\_map(map=states\_map, color="black")+  
 scale\_fill\_manual(values=pal)+  
 expand\_limits(x=states\_map$long, y=states\_map$lat)+  
 coord\_map("polyconic")+  
 labs(fill="Assault Rate\nPercentile")

# Task 2: Grouped Data (1 pt)

Continue with the flights data in library(nycflights13). Use the str( ) and View( ) functions to evaluated the data structure. To best understand how grouping of records works one needs to look at the resulting data structure.

## Task 2.1 (0.2 pts)

Describe the organization of the original flights data with respect to its variables and the nesting of the records according to year, month and day.

library(nycflights13)  
str(flights)

## Classes 'tbl\_df', 'tbl' and 'data.frame': 336776 obs. of 19 variables:  
## $ year : int 2013 2013 2013 2013 2013 2013 2013 2013 2013 2013 ...  
## $ month : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ day : int 1 1 1 1 1 1 1 1 1 1 ...  
## $ dep\_time : int 517 533 542 544 554 554 555 557 557 558 ...  
## $ sched\_dep\_time: int 515 529 540 545 600 558 600 600 600 600 ...  
## $ dep\_delay : num 2 4 2 -1 -6 -4 -5 -3 -3 -2 ...  
## $ arr\_time : int 830 850 923 1004 812 740 913 709 838 753 ...  
## $ sched\_arr\_time: int 819 830 850 1022 837 728 854 723 846 745 ...  
## $ arr\_delay : num 11 20 33 -18 -25 12 19 -14 -8 8 ...  
## $ carrier : chr "UA" "UA" "AA" "B6" ...  
## $ flight : int 1545 1714 1141 725 461 1696 507 5708 79 301 ...  
## $ tailnum : chr "N14228" "N24211" "N619AA" "N804JB" ...  
## $ origin : chr "EWR" "LGA" "JFK" "JFK" ...  
## $ dest : chr "IAH" "IAH" "MIA" "BQN" ...  
## $ air\_time : num 227 227 160 183 116 150 158 53 140 138 ...  
## $ distance : num 1400 1416 1089 1576 762 ...  
## $ hour : num 5 5 5 5 6 5 6 6 6 6 ...  
## $ minute : num 15 29 40 45 0 58 0 0 0 0 ...  
## $ time\_hour : POSIXct, format: "2013-01-01 05:00:00" "2013-01-01 05:00:00" ...

## Task 2.2 (0.5 pts)

Which **attributes** are added to the *tibble* flights\_ymd once the data are grouped? Use the grouping code:

flights\_ymd <- flights %>% group\_by(year, month, day)  
head(flights\_ymd)

How do subsequent function calls know which set of observations belongs to which group?

**The function groups those records with the same value on year, month and day into one set.**

## Task 2.3 (0.2 pts)

Describe how the grouping structure of flights\_ym changes after executing the code:

flights\_ym <- summarize(flights\_ymd, flights=n())  
str(flights\_ym)

**It summarizes the grouped table based on a different date (year, month and day), and count how many records in each set.**

## Task 2.4 (0.1 pts)

What does the ungroup function do to the data structure of flights\_new. Evaluate the output of the code:

flights\_new <- flights\_ymd %>% ungroup()

**Transform a grouped table to an ordinary one (no more groups information).**

# Task 3: Sequential, Nested or Piped Execution of Commands (1 pt)

## Task 3.1 (0.6 pts)

Create a simple example using real data and function calls of a sequence of commands. Show for your example for the three different implementations:

* Sequential:
* y <- f(x)  
   z <- g(y)
* Nested:
* z <- g(f(x))
* Piped:
* z <- x %>% f( ) %>% g( )

Each implementation should give identical results z.

x <- 16  
y <- sqrt(x)  
z <- sqrt(y)  
z

## [1] 2

z <- sqrt(sqrt(x))  
z

## [1] 2

z <- x %>% sqrt( ) %>% sqrt( )  
z

## [1] 2

## Task 3.2 (0.4 pts)

Discuss the practical advantages and disadvantages of each implementation.

**1.When you need the keep the value of Intermediate variable, could use Sequential**

**2.If you do not need the Intermediate variable, and two functions do not need many arguments, using Nested programming could save a lot of space.**

**3.The same situation with the second scenario，but two functions need plenty of input arguments, the second style would look massive in this time. So we could choose the Piped style here.**