R Noobz: Class 2
Data Cleaning

YouthFirst Lab

Class Goals

1. **More about R**
    a. *Understanding Functions*
    b. *Vectors*
    c. *Using Syntax Files*
2. **Introduction to Data Cleaning**
    a. *Data Cleaning Process/Steps*
    b. *Data Cleaning Packages: Intro to Dplyr*

More R Stuff

R Stuff: Understanding Functions

- What is a function?
  - *A piece of code written to carry out a specified task*
- Have you used any functions?
  - *Name some!!*
    - ***getwd()***
    - ***setwd()***
    - ***library()***
    - ***class()***
- Functions all have predefined argument

Function	Description
seq(*from* , *to*, *by*)	generate a sequence indices <- seq(1,10,2) #indices is c(1, 3, 5, 7, 9)

R Stuff: Understanding Functions

Practice Question #1

- Let's use the sequence function
- Create an object called "rnoobz" that contains numbers 1-100, going by 5's
- What is the sum of rnoobz?

Function	Description
seq(*from* , *to*, *by*)	generate a sequence indices <- seq(1,10,2) #indices is c(1, 3, 5, 7, 9)

R Stuff: Understanding Functions

Arguments

x	the coordinates of points in the plot. Alternatively, a single plotting structure, function or *any R object with a* `plot` *method* can be provided.
y	the y coordinates of points in the plot, *optional* if `x` is an appropriate structure.
...	Arguments to be passed to methods, such as graphical parameters (see `par`). Many methods will accept the following arguments:

`type` what type of plot should be drawn. Possible types are

- `"p"` for **p**oints,
- `"l"` for **l**ines,
- `"b"` for **b**oth,
- `"c"` for the lines part alone of `"b"`,
- `"o"` for both '**o**verplotted',
- `"h"` for '**h**istogram' like (or 'high-density') vertical lines,
- `"s"` for stair **s**teps,
- `"S"` for other **s**teps, see 'Details' below,
- `"n"` for no plotting.

All other `type` s give a warning or an error; using, e.g., `type = "punkte"` being equivalent to `type = "p"` for S compatibility. Note that some methods, e.g. `plot.factor`, do not accept this.

`main` an overall title for the plot: see `title`.

`sub` a sub title for the plot: see `title`.

`xlab` a title for the x axis: see `title`.

`ylab` a title for the y axis: see `title`.

`asp` the y/x aspect ratio, see `plot.window`.

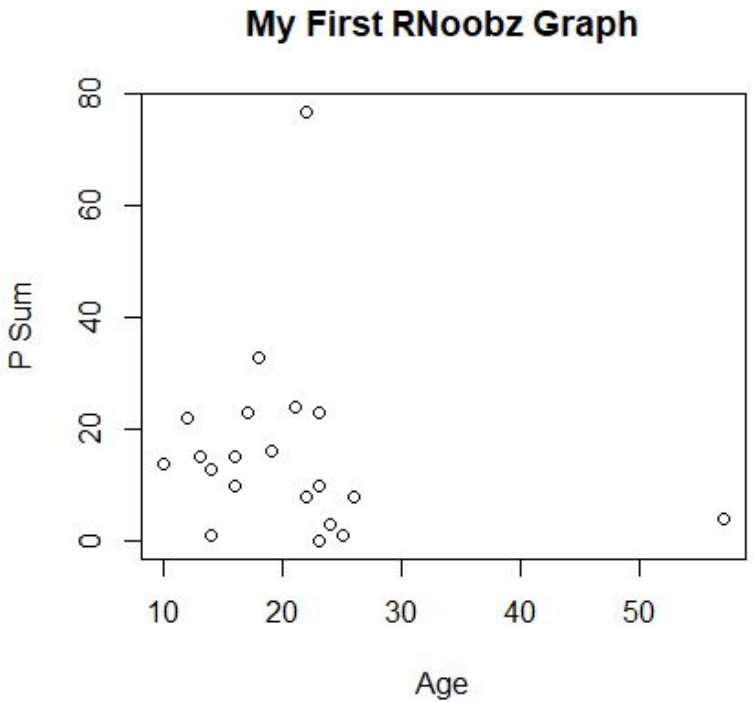
R Stuff: Understanding Functions

Practice Question #2

Can anyone reproduce this graph, using the function shown previously?

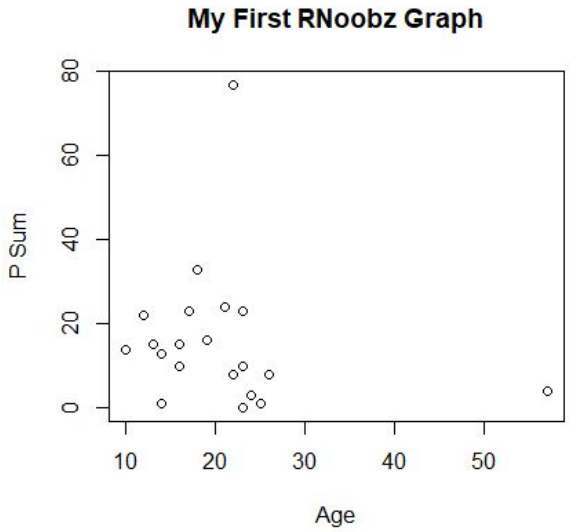
Hint:
Plotting function is:

plot(x = , y = , type = , main = , xlab = , ylab =)



R Stuff: Understanding Functions

Practice Question #2




```
plot(rnoobz2$Age, rnoobz2$P_Sum, type="p", main="My First RNoobz Graph", xlab="Age", ylab="P Sum")
```

R Stuff: Vectors

- What's a vector?
  - Vector is a basic data structure in R. It contains element of the same type. The data types can be logical, integer, double, character, complex or raw
- How do you get one tho?
  - *Vectors are generally created using the `c()` function.*
- Oh cool--why do I care tho?
  - In R you use vectors all the time!!

```
ggplot(ema, aes(x=EMVIG_M)) + geom_density(alpha=.3, color = "black", fill = "grey") + theme_classic() + labs(x = "In
Vivo Vigorous", y = "") + scale_x_continuous(limits = c(0,13), expand=c(0,0)) + scale_y_continuous(expand=c(0,0)) +
theme(axis.title.y=element_blank(),
        axis.text.y=element_blank(),
        axis.ticks.y=element_blank())
```

R Stuff: Vectors

Practice Question #3

- Create a vector called “name” including the items:
  - *Donald*
  - *Melania*
  - *Eric*
  - *Ivanka*
  - *Donald Jr.*
  - *Tiffany*
  - *Barron*
- And let’s see it!
- Any problems making this vector?

```
name <- c("Donald","Melania","Eric", "Ivanka","Donald Jr.","Tiffany","Barron")
name
```

R Stuff: Vectors

Practice Question #3

- Create a vector called “age” including the items:
  - 72
  - 48
  - 34
  - 36
  - 40
  - 24
  - 12

```
name <- c("Donald","Melania","Eric", "Ivanka","Donald Jr.","Tiffany","Barron")
name
```

```
age <- c(72, 48, 34, 36, 40, 24, 12)
age
```

R Stuff: Vectors

Practice Question #3

- Create a vector called “covfefe” including the items:
  - *88.92*
  - *21.78*
  - *63.97*
  - *57.00*
  - *1*
  - *99.99*
  - *74.25*

```
covfefe <- c(88.92, 21.78, 63.97,57.00, 1, 99.99, 74.25)
covfefe
```

R Stuff: Vectors

Practice Question #3

- Let's get super fancy!!
  - *Step 1: Let's combine these vectors!!!*
    - ***cbind()***
      - ***cbind(x1,x2,...)***
  - *Step 2: Turn it into a matrix, where there are 3 variables and 7 observations*
    - ***matrix()***
      - ***matrix(data, nrow = , ncol =)***
  - *Step 3: Name the 3 columns "Name", "Age", and "Covfefe"*
    - ***colnames(data) <- c()***
  - *Step 4: Turn this into a data frame*
    - *x <- as.data.frame(x)*
  - *Step 5: Print it out!!*

```
trump <- cbind(c(name, age, covfefe))

trump2 <- matrix(trump, nrow=7, ncol=3)

colnames(trump2) <- c("Name", "Age", "Covfefe")

trump2 <-as.data.frame(trump2)

trump2
```

```
> trump2
        Name Age Covfefe
1     Donald  72   88.92
2    Melania  48   21.78
3       Eric  34   63.97
4     Ivanka  36      57
5 Donald Jr.  40       1
6    Tiffany  24   99.99
7     Barron  12   74.25
> |
```

R Stuff: Syntax Files

- Sometimes it takes forever to write code, and its worth it to save it so you can open the file up and rerun it!
- Huzzah--you can save syntax files easily in R!
- Syntax file types:
  - *Basic → Script*

R Stuff: Syntax Files

Source on Save | Run | Source

```r
#Practice Question 1#

rnoobz <- seq(1,100,5)

sum(rnoobz)

##Practice Question 2##
rnoobz2 <- read.csv("RNoobz2.csv")

plot(rnoobz2$Age, rnoobz2$P_Sum, type="p", main="My First RNoobz Graph", xlab="Age", ylab="P Sum")

##Practice Question 3##

name <- c("Donald","Melania","Eric", "Ivanka","Donald Jr.","Tiffany","Barron")
name

age <- c(72, 48, 34, 36, 40, 24, 12)
age

covfefe <- c(88.92, 21.78, 63.97,57.00, 1, 99.99, 74.25)
covfefe

trump <- cbind(c(name, age, covfefe))
trump2 <- matrix(trump, nrow=7, ncol=3)
colnames(trump2) <- c("Name", "Age", "Covfefe")
trump2 <-as.data.frame(trump2)
trump2

```

R Stuff: Syntax Files

- Sometimes it takes forever to write code, and its worth it to save it so you can open the file up and rerun it!
- Huzzah--you can save syntax files easily in R!
- Syntax file types:
  - *Basic → Script*
  - *Cool → R Notebook*
    - *Can write code and see output below in the notebook!*

R Stuff: Syntax Files

```
---
title: "R Notebook Example"
output: html_notebook
---

This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute code within the notebook, the results appear beneath the code.

Try executing this chunk by clicking the *Run* button within the chunk or by placing your cursor inside it and pressing *Ctrl+Shift+Enter*.

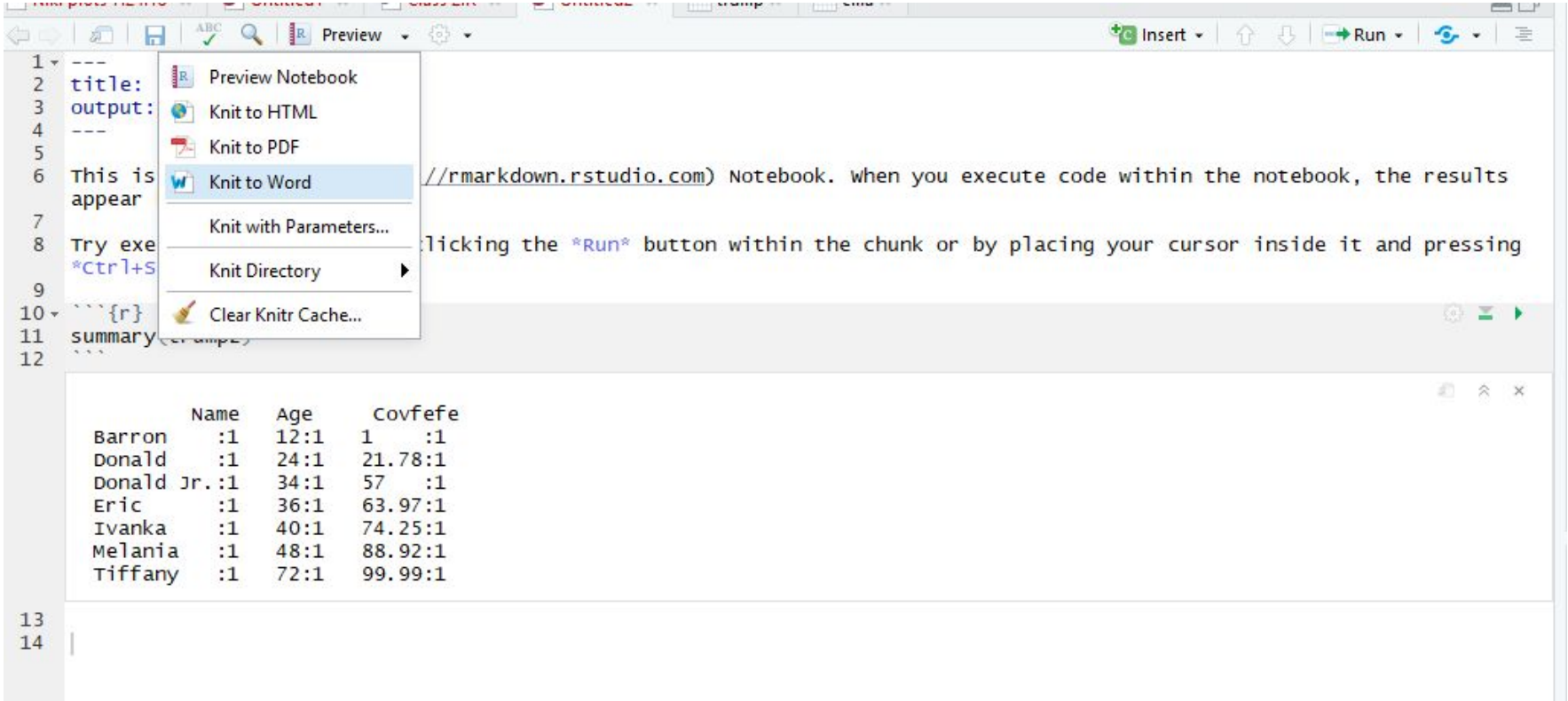
```{r}
summary(trump2)
```
```

```
        Name   Age     Covfefe
 Barron    :1   12:1   1    :1
 Donald    :1   24:1   21.78:1
 Donald Jr.:1   34:1   57   :1
 Eric      :1   36:1   63.97:1
 Ivanka    :1   40:1   74.25:1
 Melania   :1   48:1   88.92:1
 Tiffany   :1   72:1   99.99:1
```

R Stuff: Syntax Files

- Sometimes it takes forever to write code, and its worth it to save it so you can open the file up and rerun it!
- Huzzah--you can save syntax files easily in R!
- Syntax file types:
  - *Basic → Script*
  - *Cool → R Notebook*
    - *Can write code and see output below in the notebook!*
  - *Supercool → R Markdown*
    - *Can see code and output, and can save it all kinds of ways--as an HTML, PDF,*

R Stuff: Syntax Files



R Stuff: Syntax Files

- You can post rmarkdowns in github in a private repository to share w/ collaborators!

R Stuff: Syntax Files

Data Science 601 Homework #5

Pamela Rakhshan

3/21/2018

Contents


```
library("nycflights13")
library("mlbench")
library("magrittr")
library("dplyr")
```

R Stuff: Syntax Files

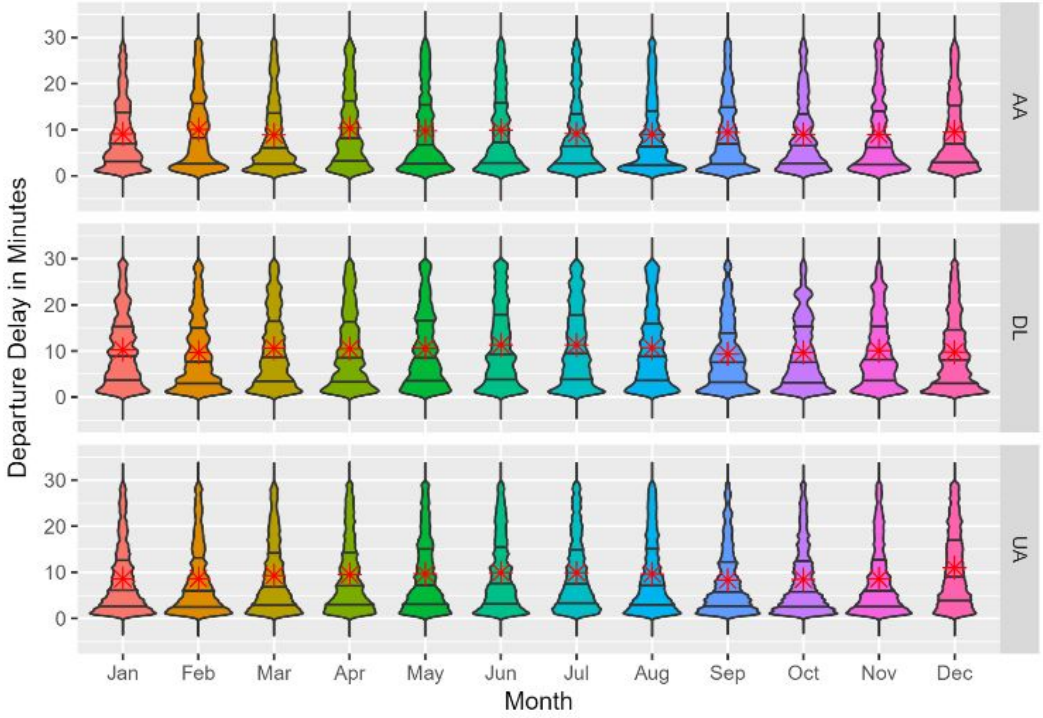
Part D

Using the output of 1.a. above (NOT 1.b.), create a new column with the total number of flights by carrier by airport. Then arrange this new column largest to smallest.

```
 table1D <- table1A %>% select(carrier,dest) %>% group_by(carrier, dest) %>% summarise(count=n())
table1D
```

```
## # A tibble: 4 x 3
## # Groups:   carrier [3]
##   carrier dest  count
##   <chr>   <chr> <int>
## 1 DL      ATL    1838
## 2 UA      ORD    1192
## 3 AA      ORD    1158
## 4 UA      ATL      42
```

R Stuff: Syntax Files



R Stuff: Syntax Files

- Or if you don't care about others seeing your work, you can post it online and people can access your file from a link!
  - *(see rpubs to get a free account)*

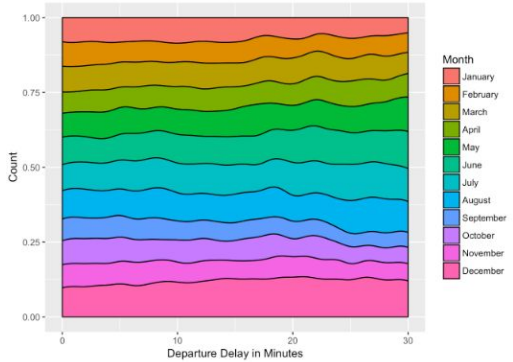
R Stuff: Syntax Files

http://rpubs.com/pjrakhshan/hw5

Part C

Start again with the output from 2.a. then pipe this information to ggplot2 and plot a stacked density plot of departure delay on the X axis, count from 0 to 1 on the Y axis and month as the fill (all carriers lumped together). Which 4 months dominate at delay = 30?

```
graph2C <- table2A %>% ggplot(aes(x = dep_delay,..count.. ,fill = as.factor(month))) + geom_density(position="fill", adjust=
3/5) + xlab(label="Departure Delay in Minutes")+ ylab(label="Count") + labs(fill="Month") + scale_fill_hue(labels=c("Januar
y", "February", "March", "April", "May", "June", "July", "August", "September", "October", "November", "December")) + xlim(0
,30) + ylim(0,1.00)
graph2C
```



The four months that comprise the greatest amount of deta at dep_delay=30 are December, August, July, and June (about 40%-50% of the sample).

Data Cleaning: A Philosophy

Pamela's Philosophy of Data Cleaning

- 95% of the work of statistical analyses is data cleaning!!
  - *Seriously!*
- It's not hard--it's just time consuming and nit picky
  - *Have a process!*
  - *Find some helpful packages*
    - *E.g. "Recoder"*
- It's not a waste of time
  - *Clean data = accurate analyses*
  - *Cleaning data → knowing your data*
  - *Noticing trends in data → cool paper ideas*

Data Cleaning Process

- Step 1: Understand structure of data
  - *str()*
  - *glimpse()*
- Step 2: Look at data
  - *summary()*
  - *plot()*
  - *head()*
  - *tail()*

Data Cleaning Process

- Step 3: Remove bad values
  - *filter()*
  - *select()*
  - *recoder()*
- Step 4: Beautify your data
  - *select()*
  - *recoder()*
  - *factor()*
  - *as.Date()*

Basic Stuff

Whaddup, data?

- Before you do ANYTHING, always always look at your data!!!
- Some good functions for this are:
  - *Basic package*
    - *str()*
    - *summary()*
    - *plot()*
    - *head()*
    - *tail()*
  - *Dplyr*
    - *glimpse()*

Whaddup, data?

Example #1

- str()
- This give you some info about the structure of your data frame

```
> str(rnoobz2)
'data.frame':   20 obs. of  7 variables:
 $ ID    : int  101 102 103 104 105 106 107 108 109 110 ...
 $ DATE  : Factor w/ 19 levels "1/1/2001","1/1/2019",..: 7 17 3 19 2 11 14 16 6 5 ...
 $ Gender: Factor w/ 2 levels "F","M": 2 1 2 2 1 1 1 2 1 2 ...
 $ Age   : int  12 14 16 14 16 17 22 24 26 57 ...
 $ P_Sum : int  22 13 15 1 10 23 8 3 8 4 ...
 $ CHR   : int  0 1 0 1 0 1 0 1 0 1 ...
 $ T2.   : logi  TRUE TRUE FALSE FALSE TRUE FALSE ...
> 
```

Whaddup, data?

Example #2

- summary()
- This give you some more info about the structure of your data frame

```
> summary(rnoobz2)
       ID                  DATE    Gender       Age             P_Sum            CHR             T2.
 Min.   :101.0   9/25/2013 : 2   F:10   Min.   :10.00   Min.   : 0.00   Min.   :0.00   Mode :logical
 1st Qu.:105.8   1/1/2001  : 1   M:10   1st Qu.:15.50   1st Qu.: 7.00   1st Qu.:0.00   FALSE:11
 Median :110.5   1/1/2019  : 1          Median :20.00   Median :13.50   Median :1.00   TRUE :9
 Mean   :110.5   1/20/2014 : 1          Mean   :20.75   Mean   :16.00   Mean   :0.55
 3rd Qu.:115.2   1/3/2016  : 1          3rd Qu.:23.00   3rd Qu.:22.25   3rd Qu.:1.00
 Max.   :120.0   10/30/2014: 1          Max.   :57.00   Max.   :77.00   Max.   :1.00
                 (Other)   :13
> |
```

Whaddup, data?

Example #3

- head()
- This give you the first few rows of the data frame

```
> head(rnoobz2)
  ID       DATE Gender Age P_Sum CHR   T2.
1 101 12/20/2018      M  12    22   0  TRUE
2 102  7/18/2017      F  14    13   1  TRUE
3 103  1/20/2014      M  16    15   0 FALSE
4 104  9/25/2013      M  14     1   1 FALSE
5 105   1/1/2019      F  16    10   0  TRUE
6 106  4/23/2017      F  17    23   1 FALSE
>
```

Whaddup, data?

Example #3

- head()
- You can specify the amount of rows you'll see by specifying n = "x"

```
> head(rnoobz2, n=3)
  ID       DATE Gender Age P_Sum CHR   T2.
1 101 12/20/2018      M  12    22   0  TRUE
2 102  7/18/2017      F  14    13   1  TRUE
3 103  1/20/2014      M  16    15   0 FALSE
>
```

Whaddup, data?

Example #4

- tail()
- Ditto head(), but last few rows!

```
> ########################
> ##Example 4##
> ########################
>
> tail(rnoobz2)
    ID       DATE Gender Age P_Sum CHR   T2.
15 115  1/3/2016      M  23     0   0 FALSE
16 116 2/24/2017      F  21    24   1 FALSE
17 117 8/10/2015      M  18    33   0  TRUE
18 118 9/25/2013      F  10    14   1 FALSE
19 119 2/12/2012      F  19    16   0 FALSE
20 120  6/6/2006      M  22    77   1  TRUE
> tail(rnoobz2, n=3)
    ID       DATE Gender Age P_Sum CHR   T2.
18 118 9/25/2013      F  10    14   1 FALSE
19 119 2/12/2012      F  19    16   0 FALSE
20 120  6/6/2006      M  22    77   1  TRUE
>
```

Whaddup, data?

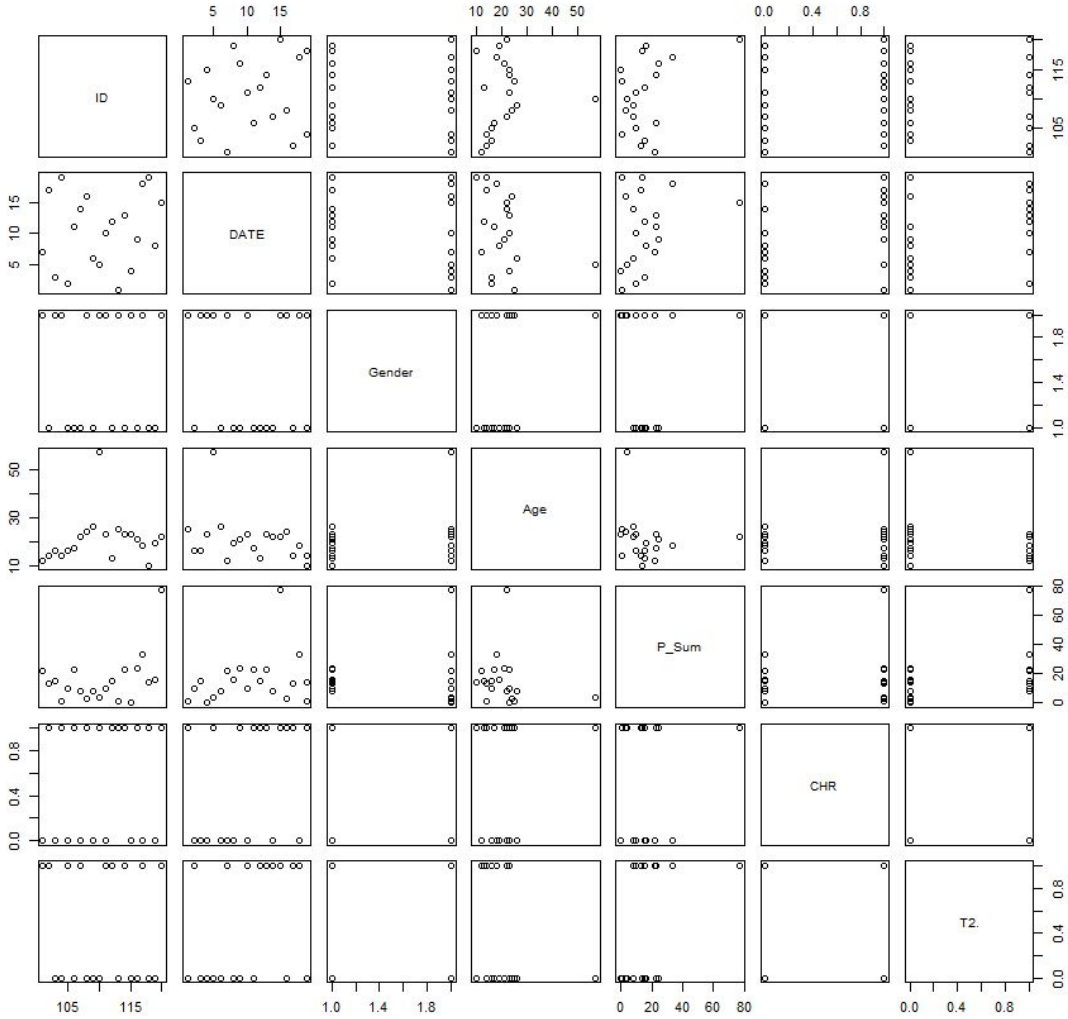
Example #5

- plot()
- Nice, easy way to visualize your data!!

Whaddup, data?

Example #5

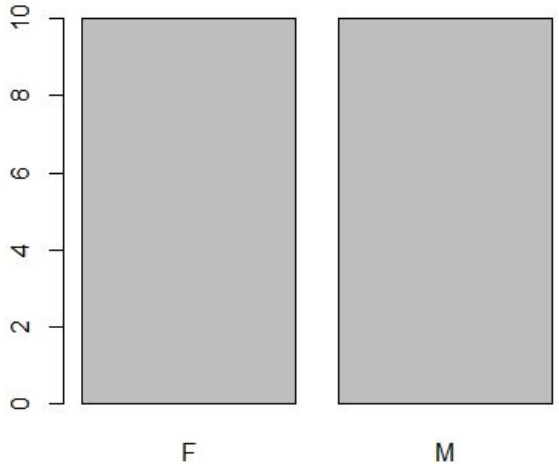
- plot(rnoobz2)


ID
DATE
Gender
Age
P_Sum
CHR
T2.

Whaddup, data?

Example #5

- plot(rnoobz2$Gender)



Whaddup, data?

Example #5

- Let’s get fancy!
- Do this:
  - *install.packages("graphics")*
  - *library(graphics)*
- One way we can look at distribution of 1 variable is through a historgram!
- Aka, **hist()**

Whaddup, data?

Example #5

- Some parts of the **hist()** function are:
  - hist(x, breaks =,
    freq = NULL, probability = !freq,
    include.lowest = TRUE, right = TRUE,
    density = NULL, angle = 45, col = NULL, border = NULL,
    main = paste("Histogram of" , xname),
    xlim = range(breaks), ylim = NULL,
    xlab = xname, ylab,
    axes = TRUE, plot = TRUE, labels = FALSE,
    nclass = NULL, warn.unused = TRUE, ...)

Whaddup, data?

Example #5

- Some parts of the **hist()** function are:

`breaks`

one of:

- a vector giving the breakpoints between histogram cells,
- a function to compute the vector of breakpoints,
- a single number giving the number of cells for the histogram,
- a character string naming an algorithm to compute the number of cells (see 'Details'),
- a function to compute the number of cells.

Whaddup, data?

Example #5

- Some parts of the **hist()** function are:

`main, xlab, ylab`	these arguments to `title` have useful defaults here.

WHADDUP, DATA?

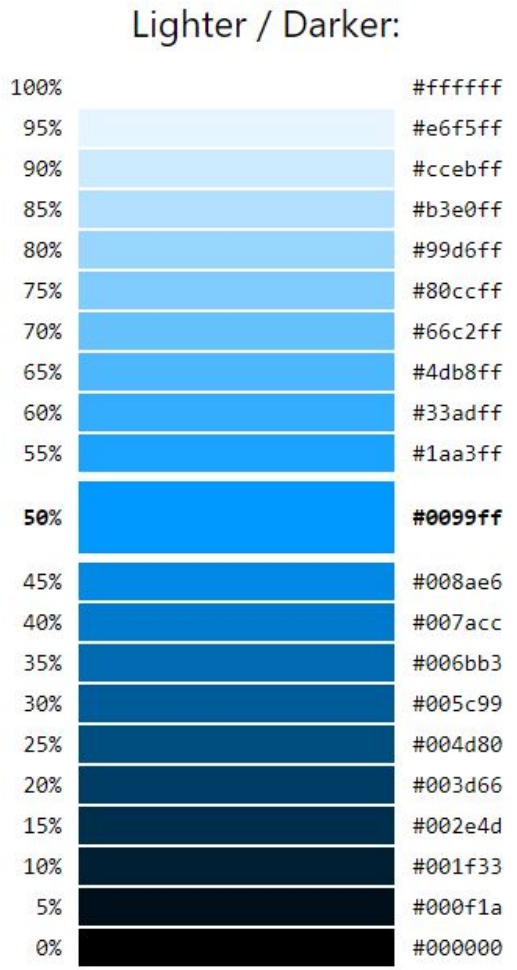
Example #5

- Some parts of the **hist()** function are:

`col`
a colour to be used to fill the bars. The default of `NULL` yields unfilled bars.

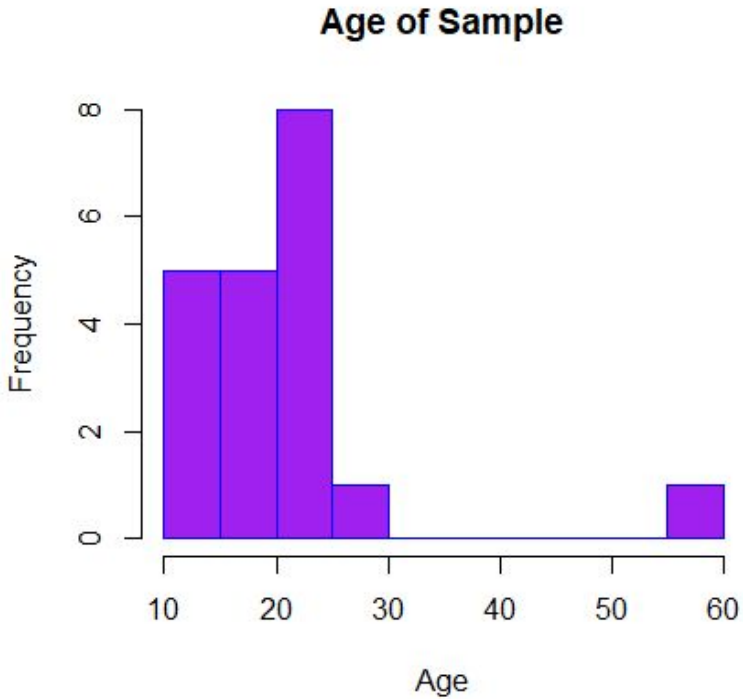
`border`
the color of the border around the bars. The default is to use the standard foreground color.

Hint: you can type in the weird #s associated with colors, or just type in the name of the most common ones in quotes (e.g. “blue”)



Whaddup, data?

Practice Question #4



Challenge!!

Can you make this graph using the hist() function?

Dplyr

DPLYR

- summarise()
- group_by()
- select()
- filter()

DPLYR: SUMMARISE()

- summarise()
  - *summarise(data, ...)*
- Say I want to know the mean age, psum, and count of my sample
  - *summarise(rnoobz2, count=n(), mean_age = mean(Age), mean_PSUM = mean(P_Sum),*

```
> summarise(rnoobz2, count = n(), mean_age = mean(Age), mean_PSUM = mean(P_Sum))
# A tibble: 1 x 3
  count mean_age mean_PSUM
  <int>    <dbl>     <dbl>
1    20     20.8        16
> |
```

Example #6

Useful Functions

- Center: `mean()`, `median()`
- Spread: `sd()`, `IQR()`, `mad()`
- Range: `min()`, `max()`, `quantile()`
- Position: `first()`, `last()`, `nth()`,
- Count: `n()`, `n_distinct()`
- Logical: `any()`, `all()`

dplyr: summarise()

Practice Question #5

- summarise()
  - *summarise(data, ...)*
- How would we get the mean, standard deviation, min, max, and count of age and psum?

Useful Functions

- Center: `mean()`, `median()`
- Spread: `sd()`, `IQR()`, `mad()`
- Range: `min()`, `max()`, `quantile()`
- Position: `first()`, `last()`, `nth()`,
- Count: `n()`, `n_distinct()`
- Logical: `any()`, `all()`

DPLYR: SUMMARISE()

Practice Question #5

```
# A tibble: 1 x 9
  count mean_age stdev_age min_age max_age mean_P stdev_P min_P max_P
  <int>    <dbl>     <dbl>   <dbl>   <dbl>  <dbl>   <dbl> <dbl> <dbl>
1    20     20.8      9.74      10      57     16    16.9      0    77
> |
```

DPLYR: GROUP_BY()

Example #7

- group_by()
  - group_by(data, variable)
  -
- group_by(.data = rnoobz2, CHR) **%>%** summarise(count = n(), mean_age = mean(Age), stdev_age = sd(Age), min_age = min(Age), max_age = max(Age), mean_P = mean(P_Sum), stdev_P = sd(P_Sum), min_P = min(P_Sum), max_P = max(P_Sum))

```
# A tibble: 2 x 10
    CHR count mean_age stdev_age min_age max_age mean_P stdev_P min_P max_P
  <int> <int>    <dbl>     <dbl>   <dbl>   <dbl>  <dbl>   <dbl> <dbl> <dbl>
1     0     9     19.4      4.42      12      26   13.6    9.54     0    33
2     1    11     21.8      12.7      10      57   18      21.5     1    77
>
>
```

DPLYR: GROUP_BY()

Practice Question #6

- group_by()
  - *Now try grouping by gender, and getting the count, mean of age, and mean of psum for the data*

DPLYR: GROUP_BY()

Practice Question #6

```
# A tibble: 2 x 4
  Gender count mean_age mean_P_Sum
  <fct>  <int>    <dbl>      <dbl>
1 F         10     18.1       15.4
2 M         10     23.4       16.6
> 
```

DPLYR: SELECT()

- select()
  - *Most used for me!*
  - *Select (data, variables)*

Example #8

```
> subset <- select(rnoobz2, ID, CHR, P_Sum)
> subset
# A tibble: 20 x 3
      ID   CHR P_Sum
   <int> <int> <int>
 1   101     0    22
 2   102     1    13
 3   103     0    15
 4   104     1     1
 5   105     0    10
 6   106     1    23
 7   107     0     8
 8   108     1     3
 9   109     0     8
10   110     1     4
11   111     0    10
12   112     1    15
13   113     1     1
14   114     1    23
15   115     0     0
16   116     1    24
17   117     0    33
18   118     1    14
19   119     0    16
20   120     1    77
> 
```

DPLYR: SELECT()

- select()
  - *Most used for me!*
  - *Select (data, variables)*

Example #8

```
> subset <- select(rnoobz2, STUDY_ID = ID, CHR_STATUS = CHR, PSUM = P_Sum)
> subset
# A tibble: 20 x 3
   STUDY_ID CHR_STATUS  PSUM
      <int>      <int> <int>
 1      101          0    22
 2      102          1    13
 3      103          0    15
 4      104          1     1
 5      105          0    10
 6      106          1    23
 7      107          0     8
 8      108          1     3
 9      109          0     8
10      110          1     4
11      111          0    10
12      112          1    15
13      113          1     1
14      114          1    23
15      115          0     0
16      116          1    24
17      117          0    33
18      118          1    14
19      119          0    16
20      120          1    77
> 
```

DPLYR: FILTER()

Example #9

- filter()
  - *filter(data, condition)*
    - *R weirdness --*
      - *sometimes when you're using R for things like conditions (if this is equal to XX), you have to use == instead of =*

```
> CHR <- filter(rnoobz2, CHR == 1)
> CHR
# A tibble: 11 x 7
      ID DATE       Gender   Age P_Sum   CHR T2.
   <int> <fct>      <fct>  <int> <int> <int> <lgl>
 1   102 7/18/2017  F         14    13     1 TRUE
 2   104 9/25/2013  M         14     1     1 FALSE
 3   106 4/23/2017  F         17    23     1 FALSE
 4   108 7/18/2009  M         24     3     1 FALSE
 5   110 10/30/2014 M         57     4     1 FALSE
 6   112 4/4/2014   F         13    15     1 TRUE
 7   113 1/1/2001   M         25     1     1 FALSE
 8   114 5/13/2016  F         23    23     1 TRUE
 9   116 2/24/2017  F         21    24     1 FALSE
10   118 9/25/2013  F         10    14     1 FALSE
11   120 6/6/2006   M         22    77     1 TRUE
>
```

DPLYR: FILTER()

Practice Question #7

- filter()
  - *Get a subsample of the data where everyone is 18 or younger, and call it "young"*
- What is the mean and standard deviation of age in this sample?

DPLYR: FILTER()

Practice Question #7

```
>
> young <- filter(rnoobz2, Age <= 18)
> young
# A tibble: 9 x 7
     ID DATE       Gender   Age P_Sum   CHR T2.
  <int> <fct>      <fct>  <int> <int> <int> <lgl>
1   101 12/20/2018 M         12    22     0 TRUE
2   102 7/18/2017  F         14    13     1 TRUE
3   103 1/20/2014  M         16    15     0 FALSE
4   104 9/25/2013  M         14     1     1 FALSE
5   105 1/1/2019   F         16    10     0 TRUE
6   106 4/23/2017  F         17    23     1 FALSE
7   112 4/4/2014   F         13    15     1 TRUE
8   117 8/10/2015  M         18    33     0 TRUE
9   118 9/25/2013  F         10    14     1 FALSE
>
> young_mean <- summarise(young, mean_age = mean(Age), sd_age = sd(Age))
> young_mean
# A tibble: 1 x 2
  mean_age sd_age
     <dbl>  <dbl>
1     14.4   2.55
>
```

Cleaning your data

Practice Question #8

- We can use these functions to clean our data!
- For example, say our study only included people ages 12-25 who were CHR +. Can you spot any issues?
- 1. Identify how many people are too old for the study
- 2. See if there are any differences in P_Sum means by age
  - *Not statistical differences, just different psums at older ages*
- 3. Exclude all people older than 18 who are not CHR positive
  - *Call this dataset "selected"*
- 4. What IDs are included in this sample?
- 5. Get the mean P_Sum value for this sample
- 6. Graph the Age Frequencies for this sample using a histogram
  - *Use blue colors from the earlier slide!*
  - *Constrain the y axis to 12-18, by axis marks of 1*

Next Class:
more data cleaning and manipulation

Homework, if you feel like it:

Data cleaning on our data!!

I will email you a de-identified copy of the SFW Data (SIPS, Demographics, and PRIME). Your task is to:

1. Remove all invalid data (e.g. a data reflecting a missing value, or missing value, outside range of IEC criteria)
2. Rename all variables using this convention
   a. P1, P2, N1, N2, P_Sum, APS, GRD, etc…
   b. Age, DOB, Gender, Race, etc…
   c. PR1, PR1A, PR2, PR2A, etc…
3. Graph P items SIPS P1- P5 and Prime Items (1-12, not distress) using histograms
4. Graph demographic spread of sample (age, gender, race, ethnicity, income)
5. Bonus Points!!
   a. Figure out how to calculate A CHR variable from APS, GRD, SPD, PSYCH, and BIPS!!

Questions?
Comments?
Thanks!!!
Go team!