

ETC3250 Business Analytics: Data Wrangling

Week 7, class 2

Souhaib Ben Taieb, Di Cook

- Example: NBA salaries
- ESPN provides basketball players' salaries for the 2013-2014 season at <http://espn.go.com/nba/salaries>

```
library(XML)
nba <- NULL
for (i in 1:11) {
  temp <- readHTMLTable(
    sprintf("http://espn.go.com/nba/salaries/_/page/%d",i))[[1]]
  nba <- rbind(nba, temp)
}
```

```
glimpse(nba)
```

```
#> Observations: 447
```

```
#> Variables: 4
```

```
#> $ RK      <fctr> 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, RK, 11, 12,
```

```
#> $ NAME    <fctr> LeBron James, SF, Mike Conley, PG, Al Horfo
```

```
#> $ TEAM    <fctr> Cleveland Cavaliers, Memphis Grizzlies, Bo
```

```
#> $ SALARY  <fctr> $30,963,450, $26,540,100, $26,540,100, $25
```

```
head(nba$SALARY)
```

```
# get rid of $ and , in salaries and convert to numeric:
```

```
gsub("[$,]", "", head(as.character(nba$SALARY)))
```

```
nba$SALARY <- as.numeric(gsub("[$,]", "",  
  as.character(nba$SALARY)))
```

```
#> [1] $30,963,450 $26,540,100 $26,540,100 $25,000,000 $24,559,380
```

```
#> 313 Levels: $16,073,140 $16,393,443 $16,663,575 $16,957,900
```

```
#> [1] "30963450" "26540100" "26540100" "25000000" "24559380"
```

```
#> Warning: NAs introduced by coercion
```

- Where does the warning come from?

Cleaning NBA salaries data: hunting the warning

```
nba %>% filter(is.na(SALARY)) %>% head()
```

```
#>   RK NAME TEAM SALARY
#> 1 RK NAME TEAM      NA
#> 2 RK NAME TEAM      NA
#> 3 RK NAME TEAM      NA
#> 4 RK NAME TEAM      NA
#> 5 RK NAME TEAM      NA
#> 6 RK NAME TEAM      NA
```

- We don't need these rows - delete all of them

```
dim(nba)
nba <- nba[-which(nba$RK=="RK"),]
dim(nba)

#> [1] 447    4
#> [1] 416    4
```

- Separate names into first, last, and position

```
nba <- nba %>%  
  mutate(NAME = as.character(nba$NAME)) %>%  
  separate(NAME, c("full_name", "position"), ",") %>%  
  separate(full_name, c("first", "last"), " ")
```

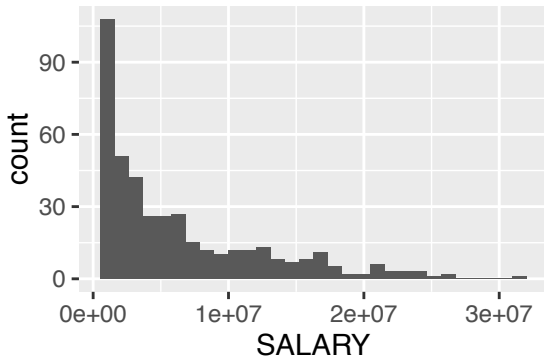


```
#>   RK   first    last position      TEAM    SALARY
#> 1  1 LeBron   James      SF Cleveland Cavaliers 30963000
#> 2  2   Mike   Conley      PG  Memphis Grizzlies 26540000
#> 3  3    Al   Horford      C   Boston Celtics 26540000
#> 4  4  Dirk Nowitzki      PF  Dallas Mavericks 25000000
#> 5  5 Carmelo Anthony      SF   New York Knicks 24550000
#> 6  6 Damian  Lillard      PG Portland Trail Blazers 24320000
```

Cleaned data ...?

- Numbers might still be wrong, but now we are in a position to check for that.

```
ggplot(data=nba, aes(x=SALARY)) + geom_histogram()
```





The Australian Electorate Commission publishes the boundaries of the electorates on their website at http://www.aec.gov.au/Electorates/gis/gis_datadownload.htm. Once the files (preferably the national files) are downloaded, unzip the file (it will build a folder with a set of files). We want to read the shapes contained in the `shp` file into R.

```
library(maptools)

# shapeFile contains the path to the shp file:
shapeFile <- "../data/vic-esri-24122010/vic 24122010.shp"
sF <- readShapeSpatial(shapeFile)
class(sF)
#> [1] "SpatialPolygonsDataFrame"
#> attr(,"package")
#> [1] "sp"
```

sF is a spatial data frame containing all of the polygons. We use the rmapshaper package available from ateucher's github page to thin the polygons while preserving the geography:

```
library(rmapshaper)
```

```
sFsmall <- ms_simplify(sF, keep=0.05) # use instead of thinner
```

`keep` indicates the percentage of points we want to keep in the polygons. 5% makes the electorate boundary still quite recognizable, but reduce the overall size of the map considerably, making it faster to plot.

We can use base graphics to plot this map:

```
plot(sFsmall)
```





A spatial polygons data frame consists of both a data set with information on each of the entities (in this case, electorates), and a set of polygons for each electorate (sometimes multiple polygons are needed, e.g. if the electorate has islands). We want to extract both of these parts.


```
nat_data <- sF@data
```

```
head(nat_data)
```

```
#>   GEODB_OID OBJECTID DIV_NUMBER ELECT_DIV NUMCCDS ACTUAL  
#> 0           1         1          1     Aston     190  92370  
#> 1           2         2          2  Ballarat     274  95003  
#> 2           3         3          3   Batman     265  96909  
#> 3           4         4          4  Bendigo     284  95729  
#> 4           5         5          5   Bruce     226  95472  
#> 5           6         6          6  Calwell     214  99031
```

```
#>   POPULATION OVER_18 AREA_SQKM SORTNAME MAP  
#> 0           0         0  98.9337  Aston Final Divisional B  
#> 1           0         0 4651.6400 Ballarat Final Divisional B  
#> 2           0         0   65.6887  Batman Final Divisional B  
#> 3           0         0 6255.0000  Bendigo Final Divisional B  
#> 4           0         0   72.6900   Bruce Final Divisional B  
#> 5           0         0  174.7130  Calwell Final Divisional B
```

```
#>   LENGTH SHAPE_AREA
```

```
#> 0  58422.89  99056594
```

```
#> 1  117555.12 1652896627
```

The row names of the data file are identifiers corresponding to the polygons
- we want to make them a separate variable:

```
nat_data$id <- row.names(nat_data)
```

Extracting the polygon information



The `fortify` function in the `ggplot2` package extracts the polygons into a data frame.

```
nat_map <- ggplot2::fortify(sFsmall)
head(nat_map)
```

#>		long	lat	order	hole	piece	id	group
#>	1	2525287	2407463	1	FALSE	1	0	0.1
#>	2	2525840	2407413	2	FALSE	1	0	0.1
#>	3	2527187	2406561	3	FALSE	1	0	0.1
#>	4	2527167	2406442	4	FALSE	1	0	0.1
#>	5	2527987	2406306	5	FALSE	1	0	0.1
#>	6	2527967	2406186	6	FALSE	1	0	0.1

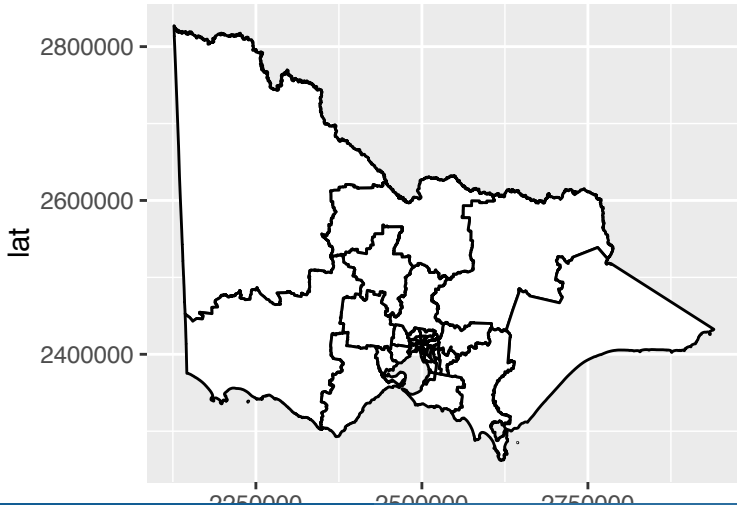
We need to make sure that `group` and `piece` are kept as factor variables – if they are allowed to be converted to numeric values, it messes things up, because as factor levels `9` and `9.0` are distinct, whereas they are not when interpreted as numbers ...

```
nat_map$group <- paste("g",nat_map$group,sep=".")
nat_map$piece <- paste("p",nat_map$piece,sep=".")
head(nat_map)
```

#>	long	lat	order	hole	piece	id	group
#> 1	2525287	2407463	1	FALSE	p.1	0	g.0.1
#> 2	2525840	2407413	2	FALSE	p.1	0	g.0.1
#> 3	2527187	2406561	3	FALSE	p.1	0	g.0.1
#> 4	2527167	2406442	4	FALSE	p.1	0	g.0.1
#> 5	2527987	2406306	5	FALSE	p.1	0	g.0.1
#> 6	2527967	2406186	6	FALSE	p.1	0	g.0.1

Plot it

```
ggplot(nat_map, aes(x=long, y=lat, group=group)) +  
  geom_polygon(fill="white", colour="black")
```



- Need to know how the missings are coded, hopefully clearly missing, treated as NA in R, not 0, or -9, or -9999, or . Recode as need be.
- Study the distribution of missing vs not missing, which will help determine how to handle them.

What ways can these affect analysis?



- If missings happen when conditions are special, eg sensor tends to stop when temperature drops below 3 degrees Celsius, estimation of model parameters may not reflect the population parameters
- Some techniques, particularly multivariate methods like many used in data mining require complete records over many variables. Just a few missing numbers can mean a lot of cases that cannot be used.

- missing completely at random (MCAR) means that values that are missing appear to be independent of everything else, just sporadically occur
- missing at random (MAR) means that missings can be dependent on other known information, eg temperature, and this information can be used to help estimate values to substitute the missing values
- missing not at random (MNAR) means that the missings are dependent on something else, but we may not have that information, which makes it impossible to appropriately estimate substitute values.

- Methods for summarising missings in a data set
- Ways to plot to examine dependence between missing vs not missing
- Imputation methods to substitute missings

```
library(MissingDataGUI)
data(tao)
MissingDataGUI(tao)
```

- eechida package vignettes
- AEC electorate polygons
- Paper on the MissingDataGUI

This work is licensed under the Creative Commons Attribution-Noncommercial 3.0 United States License. To view a copy of this license, visit <http://creativecommons.org/licenses/by-nc/3.0/us/> or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco, California, 94105, USA.