R basic with data manipulation and visualization

Dr. Uros Godnov

Contact: uros.godnov@gmail.com

Important

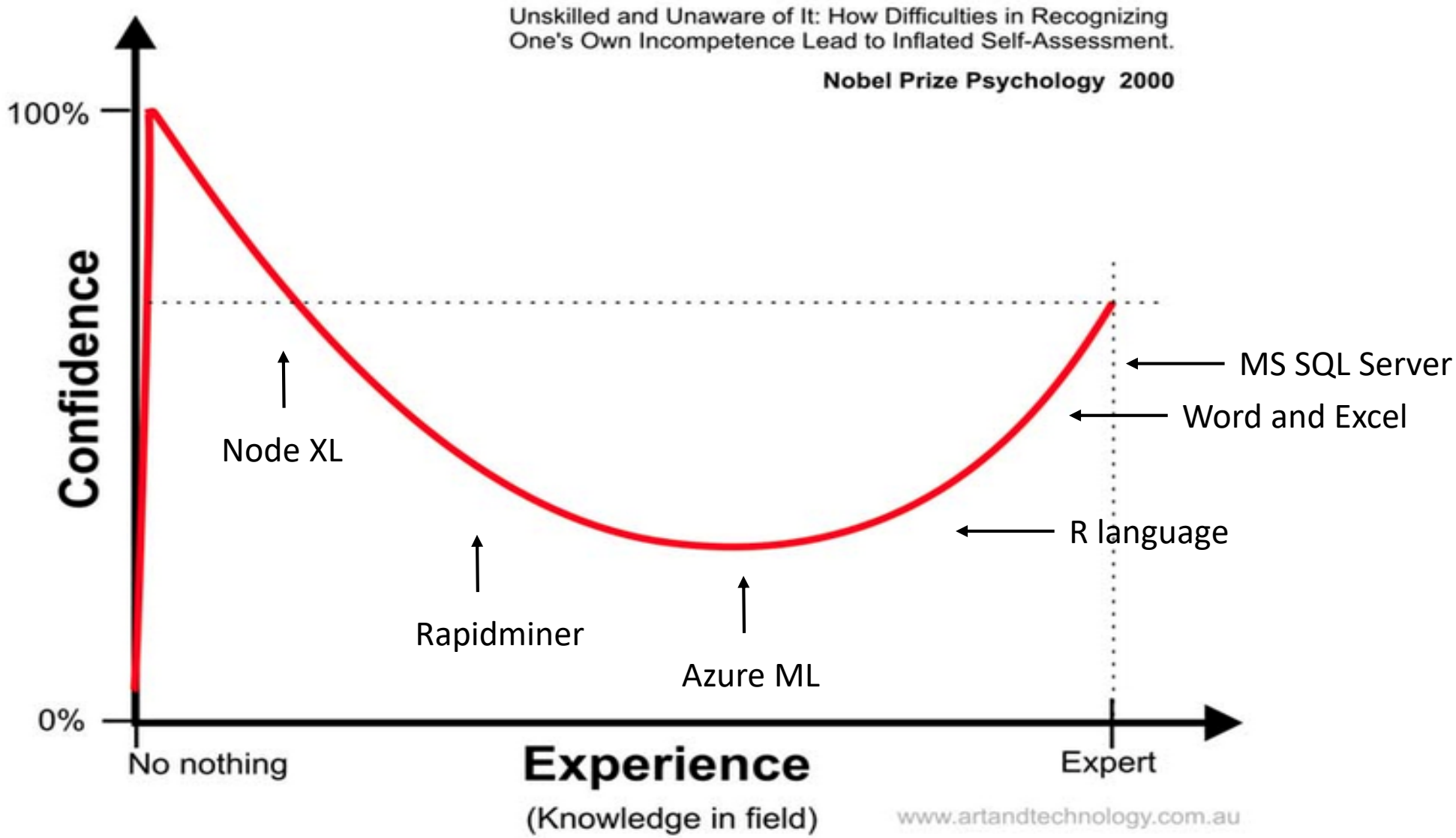
All necessary files with sample data, sample code and instructions for homework are stored on github https://github.com/urosgodnov/DataCapture.

Please download data before lesson.

Something about me

- dr. Uros Godnov
- Born 1975
- 15+ years experiences in programming, database design and BI
- Data scientist
- MCP, MCSA MS SQL 2012/2014
- Worked for Luka Koper, Steklarna Hrastnik, Svea, Simobil, UKC LJ,...
- Hobbies: bike riding

Dunning-Kruger Effect



Agenda

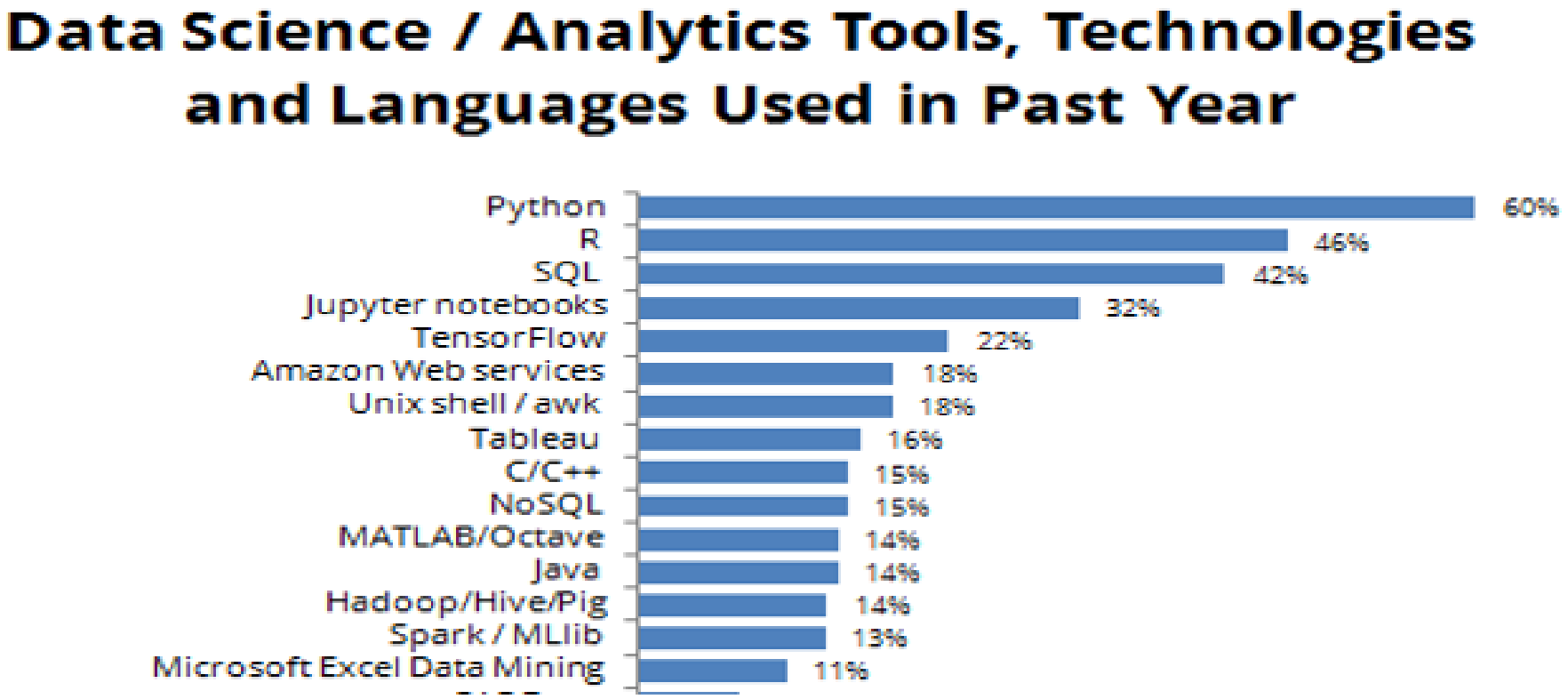
- About data science and R
- Getting around in Rstudio
- 4 objects of R
- Importing and exporting data
- Subsetting
- Manipulating with dplyr
- Plotting with ggplot2
- Data mining with R

About data science and R

Emerging job market: Data scientists

- Data scientists are more likely to be involved across the data lifecycle:
  - Acquiring new data sets: 33%
  - Parsing data sets: 29%
  - Filtering and organizing data: 40%
  - Mining data for patterns: 30%
  - Advanced algorithms to solve analytical problems: 29%
  - Representing data visually: 38%
  - Telling a story with data: 34%
  - Interacting with data dynamically: 37%
  - Making business decisions based on data: 40%

- Data for 2017



Project R

- https://cran.r-project.org/
- Packages contain functions
- On 5. 4. 2017 there were 10396 packages (<4000)
- Currently (7. 5. 2018) there are 12529 packages (<4000)
- http://www.r-pkg.org/
- IDE Rstudio:
  - Intellisense
  - Markdown
  - Notebooks
  - Debugging

    .

    .

    .
- Rstudio cheat sheets https://www.rstudio.com/resources/cheatsheets/

Getting around in Rstudio

Resources

- https://www.r-bloggers.com/
- https://cran.r-project.org/
- Coursers on coursera.org and edx.org
- Local help from packages:

  ? and ??

Demo

4 pillar objects in R

Vector

List

Matrices

Data frame/tibble

Vector

- c()
- x <- c(0.5, 0.6)
- x <- c(**TRUE**, **FALSE**)
- x <- c("a", "b", "c")
- x <- 9:29

Matrices

- Vector with dimension atrribute
- matrix()
- nrow
- ncol
- m <- matrix(nrow = 3, ncol = 3)
- every element must be the same class

List

- Very important object
- Essential part in loops
- list() and as.list() – number of parameters

Dataframe/tibble

- store tabular data
- used in a variety of statistical modeling applications
- different classes of objects in each column
- usually created by read.table() or read.csv()
- data.frame() and as.data.frame()

- Tibble is modern representation of dataframe

Additional – factors and names

- factor() – categorical data
- factors are important with lm, glm,...
- names() – you can name columns in dataframes, elements in lists,...

Exploring objects

- str() -> object‘s structure
- dim() -> object‘s dimensions

Lab

- Create a new script and name it lab1
- Create numeric vector with 100 elements
- Create list with 10 elements
- Create matrix with 10 rows and 5 columns
- Create list with 5 named elements
- Create dataframe with 4 columns and rename them

Importing and exporting data

Read.csv()

Read.csv2()

Write.csv()

Write.csv2()

-extension for financial data

Read.csv() and read.csv2()

- using wizard or manually
- can be imported from online resources
- for larger files use package reader (readxl)

- pay attention to rio package

Lab

- Search online for a sample csv data
- Download csv
- Import csv

Write.csv() and write.csv2()

Excel files

- 2 packages:

  readxl: read_xlsx

  writexl: write_xlsx

Extra – getting financial data

- Quandl package
- Quandl package returns data in a number of formats:
  'raw', 'ts', 'zoo', 'xts', 'timeSeries,
- To make more than 50 calls in a day, you‘ll need to set API
- Quandl codes at https://help.quandl.com/article/92-how-do-i-download-the-quandl-codes-of-all-the-datasets-in-a-given-database

- #bitcoin

```
quandldata = Quandl("BITFINEX/BTCUSD", start_date="2016-01-01")
plot(quandldata[,1:2])
```

Lab

- Show the price of Etherium from 1.1.2016
- Quandl code GDAX/ETH_USD

Subsetting

Dataframe

- columns:

  cars[,1] ← → cars[,c("speed")

  airquality[, c(1:3,5)]
- rows:

  cars[cars$speed==4,]

  cars[cars$speed==4 & cars$dist==2,] – and

  cars[cars$speed==4 | cars$dist>56,] – or
- both:

  airquality[airquality$Temp>75 & airquality$Wind>12, 4:5]

Dataframe

- order:

  newdata<-airquality[airquality$Temp>75 & airquality$Wind>12, 4:5]

  newdata[order(newdata$Temp),] - ascending

  newdata[order(-newdata$Temp),] – decreasing

Data manipulation

with dplyr package

Data Wrangling
with dplyr and tidyr
Cheat Sheet

RStudio

Tidy Data - A foundation for wrangling in R

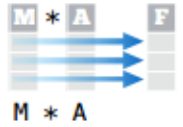
In a tidy data set:

Each **variable** is saved in its own **column**

&

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.



Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

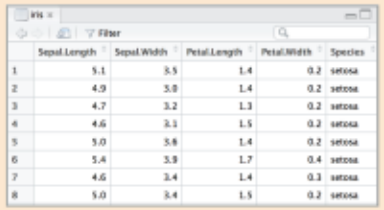
```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**

Information dense summary of tbl data.

utils::**View(iris)**

View data set in spreadsheet-like display (note capital V).



	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
1	5.1	3.5	1.4	0.2	setosa
2	4.9	3.0	1.4	0.2	setosa
3	4.7	3.2	1.3	0.2	setosa
4	4.6	3.1	1.5	0.2	setosa
5	5.0	3.6	1.4	0.2	setosa
6	5.4	3.9	1.7	0.4	setosa
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::**%>%**

Passes object on left hand side as first argument (or . argument) of function on righthand side.

```
x %>% f(y)  is the same as  f(x, y)
y %>% f(x, ., z)  is the same as  f(x, y, z )
```

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Reshaping Data - Change the layout of a data set

tidyr::**gather(cases, "year", "n", 2:4)**

Gather columns into rows.

tidyr::**spread(pollution, size, amount)**

Spread rows into columns.

tidyr::**separate(storms, date, c("y", "m", "d"))**

Separate one column into several.

tidyr::**unite(data, col, ..., sep)**

Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**

Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**

Order rows by values of a column (low to high).

dplyr::**arrange(mtcars, desc(mpg))**

Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**

Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**

Extract rows that meet logical criteria.

dplyr::**distinct(iris)**

Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**

Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**

Randomly select n rows.

dplyr::**slice(iris, 10:15)**

Select rows by position.

dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&, |, !, xor, any, all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, **contains(".")**)
Select columns whose name contains a character string.

select(iris, **ends_with("Length")**)
Select columns whose name ends with a character string.

select(iris, **everything()**)
Select every column.

select(iris, **matches(".t.")**)
Select columns whose name matches a regular expression.

select(iris, **num_range("x", 1:5)**)
Select columns named x1, x2, x3, x4, x5.

select(iris, **one_of(c("Species", "Genus"))**)
Select columns whose names are in a group of names.

select(iris, **starts_with("Sepal")**)
Select columns whose name starts with a character string.

select(iris, Sepal.Length:Petal.Width)
Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, -Species)
Select all columns except Species.

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

Summarise Data

dplyr::**summarise(iris, avg = mean(Sepal.Length))**
Summarise data into single row of values.

dplyr::**summarise_each(iris, funs(mean))**
Apply summary function to each column.

dplyr::**count(iris, Species, wt = Sepal.Length)**
Count number of rows with each unique value of variable (with or without weights).

Summarise uses **summary functions**, functions that take a vector of values and return a single value, such as:

dplyr::**first**
First value of a vector.

dplyr::**last**
Last value of a vector.

dplyr::**nth**
Nth value of a vector.

dplyr::**n**
# of values in a vector.

dplyr::**n_distinct**
# of distinct values in a vector.

IQR
IQR of a vector.

min
Minimum value in a vector.

max
Maximum value in a vector.

mean
Mean value of a vector.

median
Median value of a vector.

var
Variance of a vector.

sd
Standard deviation of a vector.

Group Data

dplyr::**group_by(iris, Species)**
Group data into rows with the same value of Species.

dplyr::**ungroup(iris)**
Remove grouping information from data frame.

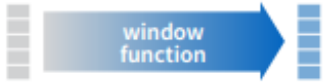
iris %>% group_by(Species) %>% summarise(…)
Compute separate summary row for each group.

Make New Variables

dplyr::**mutate(iris, sepal = Sepal.Length + Sepal. Width)**
Compute and append one or more new columns.

dplyr::**mutate_each(iris, funs(min_rank))**
Apply window function to each column.

dplyr::**transmute(iris, sepal = Sepal.Length + Sepal. Width)**
Compute one or more new columns. Drop original columns.



Mutate uses **window functions**, functions that take a vector of values and return another vector of values, such as:

dplyr::**lead**
Copy with values shifted by 1.

dplyr::**lag**
Copy with values lagged by 1.

dplyr::**dense_rank**
Ranks with no gaps.

dplyr::**min_rank**
Ranks. Ties get min rank.

dplyr::**percent_rank**
Ranks rescaled to [0, 1].

dplyr::**row_number**
Ranks. Ties got to first value.

dplyr::**ntile**
Bin vector into n buckets.

dplyr::**between**
Are values between a and b?

dplyr::**cume_dist**
Cumulative distribution.

dplyr::**cumall**
Cumulative `all`

dplyr::**cumany**
Cumulative `any`

dplyr::**cummean**
Cumulative `mean`

cumsum
Cumulative `sum`

cummax
Cumulative `max`

cummin
Cumulative `min`

cumprod
Cumulative `prod`

pmax
Element-wise `max`

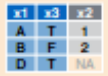
pmin
Element-wise `min`

iris %>% group_by(Species) %>% mutate(…)
Compute new variables by group.

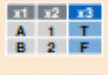
Combine Data Sets

Mutating Joins

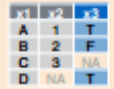
dplyr::**left_join(a, b, by = "x1")**
Join matching rows from b to a.



dplyr::**right_join(a, b, by = "x1")**
Join matching rows from a to b.



dplyr::**inner_join(a, b, by = "x1")**
Join data. Retain only rows in both sets.

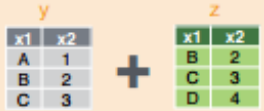


dplyr::**full_join(a, b, by = "x1")**
Join data. Retain all values, all rows.

Filtering Joins

dplyr::**semi_join(a, b, by = "x1")**
All rows in a that have a match in b.

dplyr::**anti_join(a, b, by = "x1")**
All rows in a that do not have a match in b.



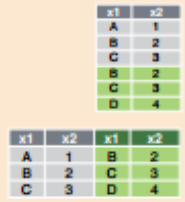
Set Operations

dplyr::**intersect(y, z)**
Rows that appear in both y and z.

dplyr::**union(y, z)**
Rows that appear in either or both y and z.

dplyr::**setdiff(y, z)**
Rows that appear in y but not z.

Binding



dplyr::**bind_rows(y, z)**
Append z to y as new rows.

dplyr::**bind_cols(y, z)**
Append z to y as new columns.
Caution: matches rows by position.

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with browseVignettes(package = c("dplyr", "tidyr")) • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

Dplyr‘s grammar

- select: return a subset of the columns of a data frame, using a flexible notation
- filter: extract a subset of rows from a data frame based on logical conditions
- arrange: reorder rows of a data frame
- rename: rename variables in a data frame

Dplyr‘s grammar

- mutate: add new variables/columns or transform existing variables
- summarise / summarize: generate summary statistics of different variables in the data frame, possibly within strata
- %>%: the “pipe” operator is used to connect multiple verb actions together into a pipeline

Dplyr‘s select

- select(data.frame, columns)
- select(data.frame, column1:column3) – selects col1, col2, col3
- select(data.frame, -(col1:col3)) – selects all columns except col1 to col3
- starts_with() and ends_with()

Dplyr‘s filter

- subsets dataframe
- filter(data.frame, condition)

Dplyr‘s arrange

- orders dataframe according to variables
- arrange(data.frame, col1)
- arrange(data.frame, desc(col1))

Dplyr‘s rename

- renames columns
- rename(dataframe, newcol1=col1,newcol2=col2)
- new name is on the left hand side

Dplyr‘s mutate/transmutate

- computes new variables
- related verb transmutate() which drops non transformed variables

Dplyr‘s group_by()

- is used to generate summary statistics from the data frame within strata defined by a variable

Dplyr‘s %>%

- pipe operator
- stringing together multiple dplyr functions in a sequence of operations
- first(x) %>% second %>% third

Dplyr‘s lag() and lead()

- lead – next element in series
- lag – previous element in series

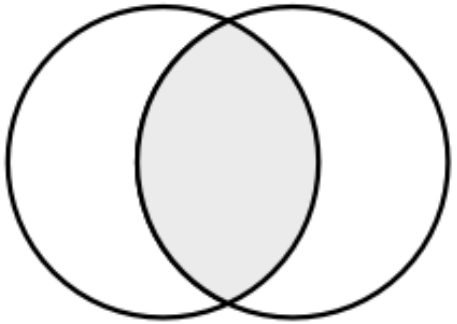
- cbind – bind results based on columns

```
x <- runif(5)
x
cbind(ahead = lead(x), x, behind = lag(x))
```

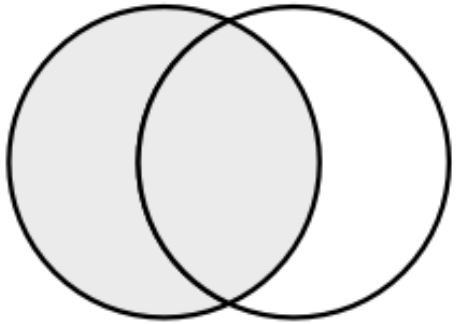
Dplyr‘s lag() and lead()

```
r<-data.frame(year=2005:2014,population=sample(14000:15000, 10, replace=T))

r<-cbind(r,lag(r$population))

names(r)<-c("year","pop","pop1")

index<-as.data.frame(r%>%group_by(year)%>%mutate(index=round(100*(1+(pop-pop1)/pop1),2)))%>%select(year,pop,index)
index
```

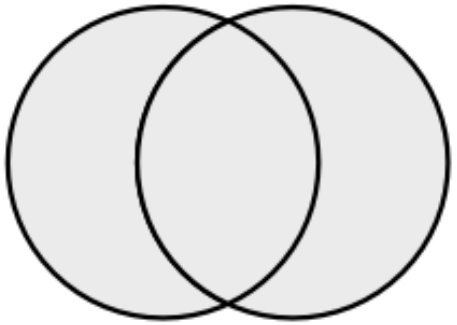
Joins



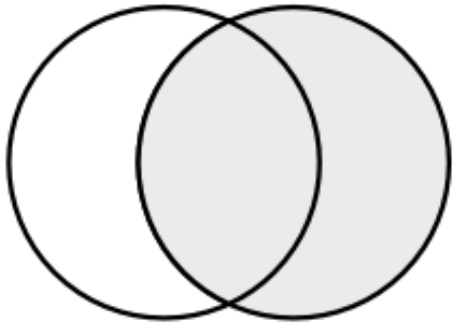
`inner_join(x, y)`



`left_join(x, y)`



`full_join(x, y)`



`right_join(x, y)`

Lab

- import Master.csv
- number of players by year of birth n()
- number of players per birthCountry (asc)
- average weight in kg per birthCountry (desc)
- help:

  select birthyear
  group by
  summarise (use function n())
  arrange

Gather and spread (tidyr)

- spread(key,value)
- gather(key, value)

```
# A tibble: 65 x 3
# Groups:   year [10]
    year month   pop
   <int> <int> <int>
 1  2005     1 14333
 2  2005     2 14745
 3  2005     4 14153
 4  2005     5 14615
 5  2005     6 42730
 6  2005     8 29600
 7  2005     9 59228
 8  2005    10 14576
 9  2005    12 14396
10  2006     1 29113
```

```
# A tibble: 10 x 13
# Groups:   year [10]
    year   `1`   `2`   `3`   `4`   `5`   `6`   `7`   `8`   `9`  `10`  `11`  `12`
   <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
 1  2005 14105 14042    NA    NA 28466    NA 14272 43339    NA 14258    NA 14984
 2  2006    NA    NA 44169 14818    NA    NA 14276 14762    NA 14130 14163    NA
 3  2007 14673 29970 14636    NA 14406 14619 29525    NA    NA 28300    NA 28240
 4  2008 14905 14484 14168 14933    NA 14889 14422    NA 14327 14347 29226 14457
 5  2009 28908    NA    NA    NA 29190 14417 14033    NA 29045 14912    NA    NA
 6  2010 14837 43624    NA    NA 14334    NA 14654 14069 28484    NA 29017    NA
 7  2011 43998    NA    NA    NA    NA    NA    NA 14405 14040    NA 14721 28450
 8  2012 44568 14178    NA 28985 14548 14214 43295 28717    NA 14169    NA 14444
 9  2013    NA    NA 14779 14217 14200    NA    NA    NA    NA 14046 14427    NA
10  2014    NA 28586 14560    NA 29003    NA    NA 29055 29251 14528    NA 14671
```

Lab

- Import norway_new_car_sales_by_model.xlsx
- Show the number of sold cars by manufacturer (rows) and years(columns)

Data presentation

with ggplot2 package

Basic object

- ggplot(data, aes(x,y))

geom_bar()

geom_point()

geom_line()

.

.

.

Adding meta data

- ggtitle()
- xlab()
- ylab()

Changing elements with theme()

- few themes in ggplot2 package
- for more install ggthemes package

Changing elements with ggThemeAssist

- addin for RStudio
- install ggThemeAssist package
- ggThemeAssistGadget(plot)

Facet_grid()

- adding 1 or 2 additional variables
- facet_grid(variable1~.)
- facet_grid(.~variable1)
- facet_grid(variable1~variable2)

geom_smooth()

- aids the eye in seeing patterns in the presence of
- method: lm, glm,...
- se: confidence interval by default TRUE

Displaying plots side by side

- package gridExtra
- grid.arrange(plot1, plot2, ncol/nrow)

Geom_bar()

- geom_bar(stat parameter)
- use identity if values are already summarized
- you can also use count
- position=dodge – side by side

- coord_flip()

- if labels are messy, use rotation:

theme(axis.text.x = element_text(angle = 60, hjust = 1))

Lab

- create stacked chart from Titanic dataset across Class, Age and Gender

Interesting tip

- How to create interactive chart

- plotly library
- use ggplotly for ggplot2 objects

R notebook

R notebook

- Basically rmarkdown
- Pdf, html, doc

- Reference: https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf
- If you want to export in pdf format, be sure to install http://miktex.org/2.9/setup