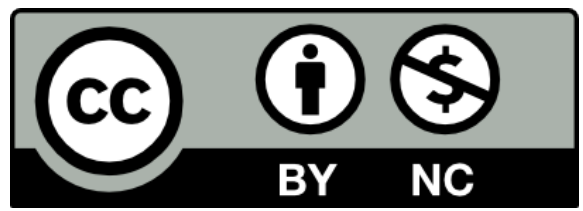
Business 4720 - Class 5

Data Management in R using Tidyverse

Joerg Evermann

Faculty of Business Administration
Memorial University of Newfoundland
jevermann@mun.ca



MEMORIAL UNIVERSITY

This Class

What You Will Learn:

- Introduction to R
- Introduction to the Tidyverse set of packages
  - ”Tidy” data
  - Manipulating & clearning data
  - Joining data
  - Summarizing and reporting data
- Data cleaning

MEMORIAL UNIVERSITY

Intro to R

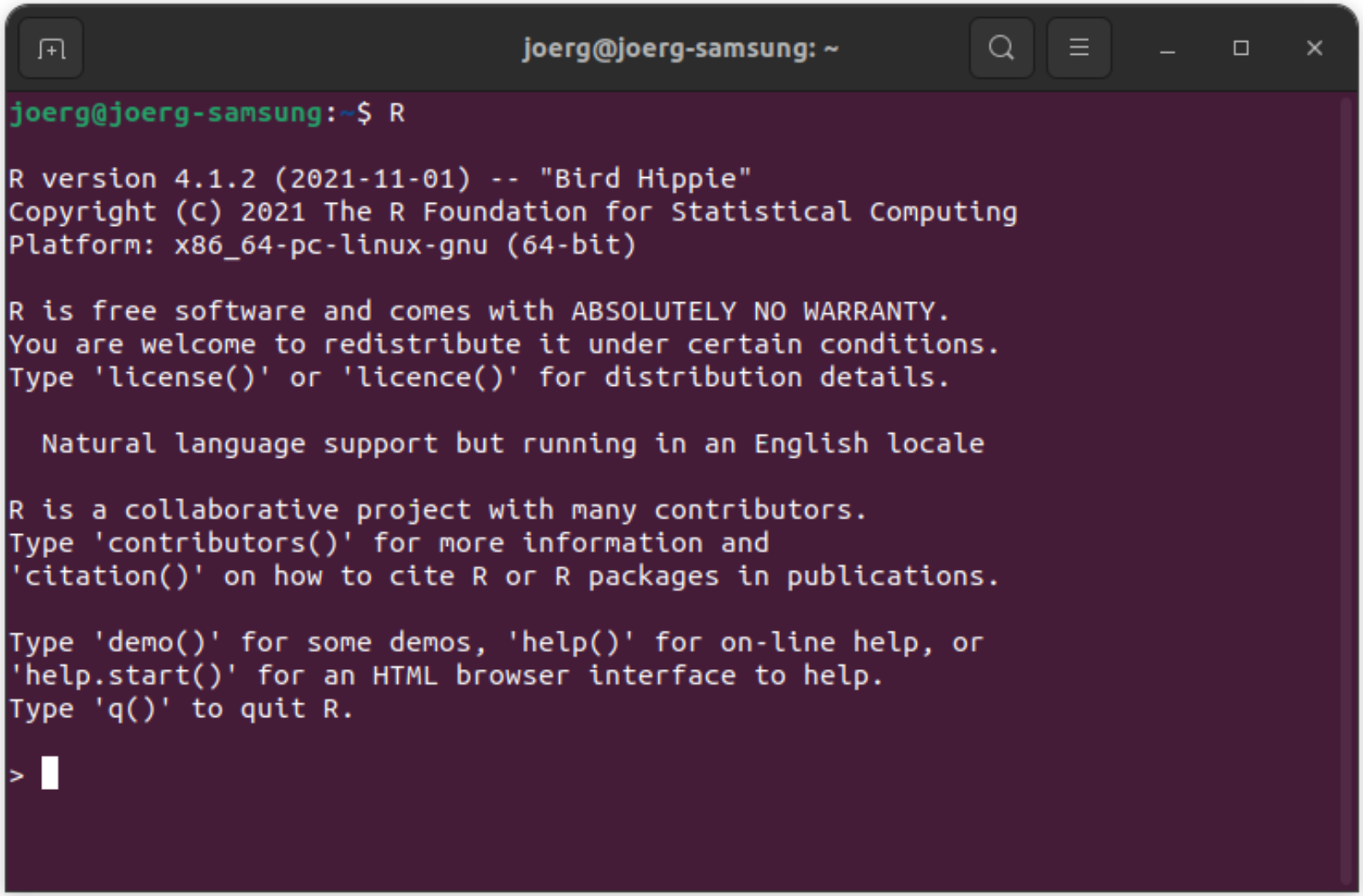
What is R?

- System for statistical analyses
- Created in 1993
- R programming language
- Open-source, cross-platform
- Widely used, popular
- Extensible, thousands of packages
- Scripting and programming

Intro Tutorial: `https://cran.r-project.org/doc/manuals/r-release/R-intro.pdf`

MEMORIAL UNIVERSITY

R


joerg@joerg-samsung: ~
joerg@joerg-samsung:~$ R
R version 4.1.2 (2021-11-01) -- "Bird Hippie"
Copyright (C) 2021 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)
R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>

My First R Command

R can do math:

```
> 1+1
[1] 2
```

R knows variables:

```
> a <- 3
> b <- 2
> print(a * b)
[1] 6
```

Note: You can also assign using "="

MEMORIAL UNIVERSITY

R Basics

R does vector math:

```
> v <- c(1, 2, 3, 4)
> v*3
[1]  3  6  9 12
```

Note: No print statement needed in interactive mode

```
> s <- seq(0, 6, by=.5)
> print(s)
> r <- rep(3.5, 5)
> print(r)
```

MEMORIAL UNIVERSITY

R Basics [cont'd]

More basic functions on vectors:

```
> length(v)
> max(v)
> min(v)
> sqrt(vv)
> var(v)
> sd(v)
> vv <- c(v, c(7, 8, 9), v)
> print(vv)
```

R Basics [cont'd]

Vector indexing:

```
> vv[vv < 5]
> vv[vv < 5] <- vv[vv < 5] + 5
> vv[3:7]
> vv[-(3:7)]
```

R Basics [cont'd]

R knows if its not a number:

```
> 2 / 0
[1] Inf
> 0 / 0
[1] NaN
```

MEMORIAL UNIVERSITY

R Basics [cont'd]

R knows NA:

```
> v[3] <- NA
> v*3
[1]  3  6 NA 12
> is.na(v)
[1] FALSE FALSE  TRUE FALSE
> sum(v)
[1] NA
```

Note: R indexes start at 1!
Note: TRUE and FALSE can be abbreviated T and F

MEMORIAL UNIVERSITY

R Basics [cont'd]

R knows boolean logic:

```
> TRUE & FALSE
FALSE
> TRUE | FALSE
TRUE
```

R know character strings:

```
> label1 = 'I Love R'
> label2 = 'and BUSI 4760'
> paste(label1, label2, sep=' ')
> strsplit('Hello World! My first string', ' ')
```

MEMORIAL UNIVERSITY

R Basics [cont'd]

R types and type coercion:

```
> is.numeric(vv)
> is.integer(vv)
> mode(vv)
> as.character(vv)
> is.character(as.character(vv))
> as.factor(as.character(vv))
> levels(as.factor(as.character(vv)))
```

MEMORIAL UNIVERSITY

R Environment

R can manipulate its objects ("workspace")

```
> ls()
[1] "a"          "b"          "v"
> rm(v)
> ls()
[1] "a"          "b"
```

R Basics [cont'd]

R knows Regex:

```
> grep('^([0-9]{3})[ -]?[0-9]{3}[ -]?[0-9]{4}$',
    c('709 864 5000', 'abc def 9999', '709-865-5000'))
[1] 1 3
> grep('[A-V][0-9][A-V] [0-9][A-V][0-9]',
    c('A0P 1L0', '0AB L2K', 'A0X 1Z0'))
[1] 1
```

R knows Levenshtein distances:

```
> agrep('apple',
    c('apricot', 'banana', 'grape', 'pineapple'),
    max.distance=3)
[1] 1 3 4
```

MEMORIAL UNIVERSITY

R Environment [cont'd]

R can help:

```
> help()
> help(lm)
> ?lm
> ??lm
> help.start()
```

R Environment [cont'd]

R knows files and directories:

```
> getwd()
[1] "/home/busi4720"
> setwd('DataSets')
> getwd()
[1] "/home/busi4720/DataSets"
> list.files()
```

Note: Strings may be enclosed in double or single quotes.

MEMORIAL UNIVERSITY

R Environment [cont'd]

R uses packages:

```
> search()
> library(matrixcalc)
> search()
> library()
> install.packages('lavaan')
> library()
> installed.packages()
```

R can read your command files:

```
> source('MyFirstScript.R')
```

Note: Sourcing a script turns off auto-printing, you must use explicit `print()` commands

MEMORIAL UNIVERSITY

R Environment [cont'd]

Say goodbye to R:

```
> quit()
```

R stores its **workspace** in each directory in a file called ".RData" and will read it when restarted. R stores its **command history** in each directory in a file called ".Rhsitory" and will read it when restarted.

MEMORIAL UNIVERSITY

R Arrays and Matrices

Arrays and matrices have dimensions:

```
> a <- array(1:20, dim=c(4,5))
> a
     [,1] [,2] [,3] [,4] [,5]
[1,]    1    5    9   13   17
[2,]    2    6   10   14   18
[3,]    3    7   11   15   19
[4,]    4    8   12   16   20
> a[,2]
> a[,2:4]
> a[3,2:4]
> a[3:1,2:4]
> i <- array(c(1:3,3:1), dim=c(3,2))
> a[i] <- 0
> a
```

R Arrays and Matrices [cont'd]

```
> b <- matrix(20:1, nrow=5, byrow=T)
> b
     [,1] [,2] [,3] [,4]
[1,]   20   19   18   17
[2,]   16   15   14   13
[3,]   12   11   10    9
[4,]    8    7    6    5
[5,]    4    3    2    1
> is.matrix(b)
> is.matrix(a)
> t(b)
> cbind(a, t(b))
> rbind(t(a), b)
```

R Lists

R knows lists:

```
> l <- list('a', 3, 'b', 2, TRUE)
> l[[2]]
> l[2]
> is.list(l)
> is.list(l[[2]])
> is.list(l[2])
> as.list(vv)
```

Vectors are not lists: Lists can contain elements of different types, vectors cannot:

```
> c(3, 'a', TRUE)
> c(3, FALSE, TRUE)
```

MEMORIAL UNIVERSITY

Data Frames

A dataframe is a special type of list:

```
> x <- rnorm(50)
> y <- 2*x + rnorm(50)
> data <- data.frame(x, y)
> colnames(data)
> colnames(data) <- c('Pred', 'Crit')
> nrow(data)
> ncol(data)
> data$Pred
> data$Crit
> summary(data)
> head(data)
> tail(data)
> cov(data)
```

MEMORIAL UNIVERSITY

Data Frames [cont'd]

Writing data to CSV:

```
> write.csv(data, 'data.csv', row.names=FALSE)
```

Reading data from CSV:

```
> new.data <- read.csv('data.csv')
> colnames(new.data)
```

MEMORIAL UNIVERSITY

Tips for Working with R

- Use the up-arrow key to retrieve earlier commands
- The `history()` function shows your command history
- Use a notepad app to assemble your commands, then copy/paste to R
- Use a notepad app for your results, copy/paste from R
- The Ubuntu terminal window uses SHIFT-CTRL-X, SHIFT-CTRL-C, SHIFT-CTRL-V
- Use multiple R windows (e.g. one for executing commands, one for reading help documentation or listing files)
- Don't update packages in the middle of a project
- Ensure you have a *repeatable, automatable script* for your entire data analysis at the end of a project

MEMORIAL UNIVERSITY

Hands-On Exercise

1. Consider the last five digits of your student ID and label those digits A–E
   - Example: Student ID=12453 $\Rightarrow$ A=1, B=2, C=4, D=5, E=3
2. Create a data frame with $(A + 1)$ columns and $10 * (B + 1)$ rows of random numbers with mean of C and standard deviation of $(D + 1)$ (use the `rnorm()` function)
3. Name the columns with letters from "A"–"K"
4. "Clip" the values so that all values lie between $-(E + 1)$ and $+(E + 1)$
5. Summarize the data and print the pairwise covariance matrix of the variables in the data frame
6. Save the data frame in a CSV file using your first name as file name (file ending '.csv')
7. Save your script in an R file using your first name as file name (file ending '.R')

MEMORIAL UNIVERSITY

Tidyverse

```
> library(tidyverse)
── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
✔ dplyr     1.1.3     ✔ readr     2.1.4
✔ forcats   1.0.0     ✔ stringr   1.5.0
✔ ggplot2   3.4.4     ✔ tibble    3.2.1
✔ lubridate 1.9.3     ✔ tidyr     1.3.0
✔ purrr     1.0.2
── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
✖ dplyr::filter() masks stats::filter()
✖ dplyr::lag()    masks stats::lag()
ℹ Use the conflicted package to force all conflicts to become errors
>
```

Intro books: `https://r4ds.hadley.nz/`

MEMORIAL UNIVERSITY

Tidyverse

dplyr	Manipulate data
forcats	Work with categorical variables (factors)
ggplot2	Grammar of Graphics
lubridate	Date and time parsing and arithmetic
purrr	Functional programming
readr	Read files in various formats
stringr	Work with character strings
tibble	A tibble is better than a table
tidyr	Make data tidy

MEMORIAL UNIVERSITY

Pagila Database in R

Read into a tibble:

```
rentals <- read_csv('rentals.csv')
head(rentals)
summary(rentals)
```

Fix the column datatypes:

```
attach(rentals)
rating <- as.factor(rating)
language <- as.factor(language)
customer_address <- as.integer(customer_address)
customer_store <- as.integer(customer_store)
rental_staff <- as.integer(rental_staff)
payment_staff <- as.integer(payment_staff)
rental_duration <- as.integer(rental_duration)
detach(rentals)
summary(rentals)
```

MEMORIAL UNIVERSITY

Pagila Database in R [cont'd]

Examine the NA's:

```
rentals |>
  filter(if_any(everything(), is.na)) |>
  select(last_name, rental_date, return_date,
         title, amount) |>
  print(n=Inf, width=Inf)
```

Interpretation:

- Some films have not been rented
- Some rentals have not been returned

Notes:

- The pipe symbol (R native pipes `|>` or magrittr pipes `%>%`)
- No quoting of column names

MEMORIAL UNIVERSITY

Pagila Database in R [cont'd]

Find all films and the actors that appeared in them, ordered by film category and year, for those films that are rated PG:

```
actors <- read_csv('actors.categories.csv')

rentals |>
    full_join(actors,
        by='title',
        suffix=c('_customer', '_actor'),
        relationship='many-to-many') |>
    filter(rating == 'PG') |>
    mutate(actor =
        paste(last_name_actor, ', ',
        first_name_actor, sep='')) |>
    rename(year=release_year) |>
    select(actor, title, category, year) |>
    distinct(actor, title, category, year) |>
    group_by(category, year, title) |>
    nest() |>
    arrange(category, year, title) |>
    relocate(category, year, title) |>
    print(n=Inf, width=Inf)
```

Summary of DPlyr ”Verbs” so far

full_join	outer join (also left_join, inner_join, right_join)
filter	filters by row
select	selects columns to retain
mutate	creates new columns
rename	renames columns
distinct	finds unique values
group_by	groups data
nest	nests data, tibbles in tibbles
arrange	sorts data rows
relocate	moves data columns
print	prints a tibble

MEMORIAL UNIVERSITY

Pagila Database in R [cont'd]

Find the most popular actors in the rentals in each city:

```
addresses <- read_csv('addresses.csv')
addresses$phone <- as.character(addresses$phone)

rentals |>
   inner_join(addresses,
       by=c('customer_address'='address_id')) |>
   inner_join(actors,
       by='title',
       suffix=c('_customer', '_actor'),
       relationship='many-to-many') |>
   mutate(actor =
       paste(last_name_actor, ', ',
       first_name_actor, sep='')) |>
   group_by(city, actor) |>
   summarize(count=n()) |>
   mutate(ranking = min_rank(desc(count))) |>
   filter(ranking < 4) |>
   arrange(city, ranking, actor) |>
   print(n=25)
```

Note: Use `rank()` to break ties, `dense_rank()` for no gaps

MEMORIAL UNIVERSITY

Pagila Database in R [cont'd]

Find the customers who spend the most on rentals, with their phone numbers and cities, and the number of rentals with the higest total rental payments for each category grouped by rental duration.

```
full_data <-
   rentals |>
     inner_join(addresses,
       by=c('customer_address'='address_id')) |>
     inner_join(actors,
       by='title',
       suffix=c('_customer', '_actor'),
       relationship='many-to-many')
```

MEMORIAL UNIVERSITY

Pagila Database in R [cont'd]

```
full_data |>
   mutate(customer=
     paste(first_name_customer, last_name_customer)) |>
   select(customer, amount, rental_duration,
          category, phone, city) |>
   group_by(category, rental_duration, customer ) |>
   mutate(payments=sum(amount), num_rentals=n()) |>
   select(-amount) |>
   group_by(category, rental_duration) |>
   mutate(ranking = min_rank(desc(payments))) |>
   slice(which.min(ranking)) |>
   print(n=Inf, width=Inf)
```

- No `summarize()`
- "Negative" `select()`
- Multiple `group_by()`
- Uses `slice()`

MEMORIAL UNIVERSITY

Pagila Database in R [cont'd]

Get the total rental revenue, number of rentals, and the mean and standard deviation of the rental amounts for each country.

```
full_data |>
  group_by(country) |>
  summarize(revenue=sum(amount),
            numrentals=n(),
            mean_amount=mean(amount),
            sd_amount=sd(amount)) |>
  arrange(desc(mean_amount),
          desc(revenue)) |>
  print(n=Inf, width=Inf)
```

MEMORIAL UNIVERSITY

Pagila Database in R [cont'd]

Get the top 5 and the bottom 5 grossing customers for each quarter.

```
full_data |>
  mutate(customer=
   paste(first_name_customer,last_name_customer)) |>
  mutate(q=
   as.character(quarter(rental_date, with_year=T))) |>
  select(customer, q, amount, rental_date) |>
  group_by(q, customer) |>
  mutate(payments=sum(amount)) |>
  select(-amount) |>
  distinct(customer, q, payments) |>
  group_by(q) |>
  mutate(rank_top = min_rank(desc(payments))) |>
  mutate(rank_bot = min_rank(payments)) |>
  filter(rank_top < 6 | rank_bot < 6) |>
```

MEMORIAL UNIVERSITY

Pagila Database in R [cont'd]

Continued from previous slide . . .

```
arrange(q, desc(payments)) |>
relocate(q, customer, payments,
             rank_top, rank_bot) |>
print(n=Inf, width=Inf)
```

- No `summarize()`
- Uses `quarter()` function from package `lubridate`
- Uses `filter()` instead of slice `slice()`

MEMORIAL UNIVERSITY

Pagila Database in R [cont'd]

Find the set of film titles by rental customer and the total number rentals for each customer

```
full_data |>
  mutate(customer=
    paste(first_name_customer,last_name_customer)) |>
  select(customer, title) |>
  nest(titles=title) |>
  rowwise() |>
  mutate(rentals=nrow(titles)) |>
  mutate(unique_titles=list(distinct(titles))) |>
  select(-titles) |>
  arrange(customer)
```

- Work with nested data using `nest` and `rowwise`

MEMORIAL UNIVERSITY

Hands-On Exercises

1. Find all films with a rating of 'PG'
2. List all customers who live in Canada (with their address)
3. Find the average *actual* rental duration for all films
   - This requires date arithmetic, use the `lubridate` package
4. Find the average overdue time for each customer
   - This requires date arithmetic, use the `lubridate` package
5. List all films that have never been rented
6. List the names of actors who have played in more than 15 films

MEMORIAL UNIVERSITY

R knows SQL

The `sqldf` Package

- Set up an in-memory SQLite database (or use existing database connection)
- Move dataframes to database tables
- Run SQL query against database
- Move result set to R dataframe
- Tear down the in-memory database (optional)

Example

```
library(sqldf)
result_df <-
   sqldf('select distinct(title) from full_data')
```

MEMORIAL UNIVERSITY

SQL Databases versus R/Tidyverse

Consider:

- **Size of data**: R is memory limited, RDBMS scale massively larger
- **Access speed**: RDBMS have sophisticated indexes and query planners
- **Currency**: Operational system RDBMS has live data
- **Transactions**: RDBMS ensure consistent views of data across multi-user, concurrent updates
- **Impact**: Queries impact transaction processing (updates of data) performance in RDBMS
- **Tools**: R has tools for statistical analysis and visualization, beyond mere reporting

MEMORIAL UNIVERSITY

SQL Databases versus R/Tidyverse [cont'd]

Recommendations:

- Do not "hit" operational RDBMS for heavy-weight or frequent analytics
- Regularly export consistent data from RDBMS
- Use separate in-memory or on-disk RDBMS for analytics (e.g. with `sqldf`) if desired/required
- If size of data is large, consider distributed tools such as Hadoop/Spark

MEMORIAL UNIVERSITY

Data Cleaning

Overview

- Critical step in the data analysis process
- Identification and rectification of errors and inconsistencies
- Improve data quality

MEMORIAL UNIVERSITY

Data Cleaning

- Critical step in the data analysis process
- Identify and ”fix” errors and inconsistencies
- Improve data quality

Activities

1. **Auditing**: Identify anomalies and inconsistencies
2. **Validation**: Ensure data conforms to rules and constraints
3. **Cleaning:** Transform and correct data
4. **Duplicate Removal:** Ensure uniqueness of data.
5. **Harmonization:** Merge datasets from different sources and ensure consistent formats and scales.
6. **Standardization:** Bring data into a standard format.
7. **Quality Assessment:** Ensure cleaning has been effective.

MEMORIAL UNIVERSITY

Data Validation

- Coding/serialization rules, e.g. with Regex
  - **Example** Are all phone numbers of the format:
    ```
    ^([0-9]{3})[ -]?[0-9]{3}[ -]?[0-9]{4}$
    ```
- Data type constraints
  - *Example*: Are all sales prices numbers?
- Range constraints
  - *Examples*: Are prices > 0? Are sales number < 1000?
- Cross-field validation
  - *Example*: If province is NL, then area code must be 709
- Uniformity of measures/scales
  - *Example*: All weights must be in kg, not pounds

MEMORIAL UNIVERSITY

Data Validation [cont'd]

- Uniqueness constraints
  - Real duplicates and synonyms
  - *Example*: Rebekah Uqi Williams
    (Commissioner of Nunavut (2020–2021)
  - **Abbreviations**:
    Rebekah U. Williams; Rebekah Williams, R.U. Williams
  - **Order**:
    Williams, Rebekah Uqi; Williams, Rebekah U.; Williams, R.
  - **Spelling**:
    Rebekah; Rebecca; Rebeccah; Rebeckah; Rebecka
  - **Misspellings**:
    Reebkah, Rebkah, Wililams, Willaims, . . .
  - . . .

MEMORIAL UNIVERSITY

Data cleaning

- **Data Transformation,** into proper format or structure.
  - One row for each observation, case, event, ...
  - Requires case or event identifiers
- **Data Imputation,** replacing missing values with estimated or default values, or removing missing values
  - Different meanings of missing values
  - Removal may bias data
  - Estimating values may be error-prone
- **Data Correction,** or removal of erroneous data.
  - Requires access to correct data

MEMORIAL UNIVERSITY

Data Cleaning

Important

Cleaning, transformation, and correction of data is *subjective* and requires *expert knowledge* of the data, the validation rules, the metadata, and the application domain.

The 80/20 Rule of Data Science

Cleaning, transformation, and correction of data takes 80% of of the time, data analysis takes 20% of the time.

MEMORIAL UNIVERSITY