Welcome to the NDACAN Summer training series!

- August 7th, 2019 - Strategies for Managing Data
- Presenter: Frank Edwards, Ph.D.
- The session will begin at 12pm.
- This session is being recorded.

NDACAN Summer Training series

National Data Archive on Child Abuse and Neglect

Bronfenbrenner Center for Translational Research

Cornell University

NDACAN Summer Training series schedule

- July 17th, 2019 - Introduction to NDACAN
- July 24th, 2019 - Overview of NCANDS Data
- July 31st, 2019 - Overview of AFCARS and NYTD Data
- **August 7th, 2019 - Strategies for Managing Data**
- August 14th, 2019 - Linking NCANDS, AFCARS, and NYTD Data
- August 21st, 2019 - Concluding Session

Session Overview

- Secondary data analysis requires a high level of data management skills, such as reshaping and collapsing data.
- This is especially true for larger datasets, such as the NCANDS dataset, where running code can be time and computing power intensive.
- This training should help you increase your data management skills and ease some of the difficulties in conducting secondary data analysis with large administrative datasets.

Strategies for managing data

Frank Edwards

Session Overview

- Use the R statistical programming language and sql-like commands to:
  - Aggregate (summarize) data to geographic or time units of analysis
  - Reshape data (wide, long)
  - Append variables or observations to existing data
  - Merge (join) datasets
  - Draw a random sample

Before we begin

- We use the free and open source R statistical programming language, and I use the RStudio IDE.
- You can install R from cran-rproject.org, and RStudio from rstudio.com
- I also use the tidyverse packages for data manipulation. Run install.packages('tidyverse') from the R console to install them
- These techniques are adapted from SQL principals, and share syntax and theory with SQL data management and wrangling principals
- These demos use AFCARS 2017 child file, with identifying variables removed

Set up

```
library(tidyverse)

afcars_17<-read_tsv("FC2017v2.tab")


afcars_17<-afcars_17%>%
  select(St, SEX, AgeAtEnd, InAtEnd, Entered)%>%
  filter(!(is.na(SEX)))
```

What are the natural groupings in this data?

St	SEX	AgeAtEnd	InAtEnd	Entered
AL	1	5	1	1
AL	1	10	1	1
AL	1	0	1	1
AL	2	4	1	1
AL	1	1	1	1
AL	2	3	1	1

Summarizing the data

```
table1<-afcars_17%>%
   summarise(MeanAge = mean(AgeAtEnd),
             PctMale = sum(SEX==1,
na.rm=TRUE)/n(),
             TotalEntries=sum(Entered),
             Caseload = sum(InAtEnd))
```

MeanAge	PctMale	TotalEntries	Caseload
8.098175	0.5157354	269732	442681

Grouping and summarizing: by sex

```
table1<-afcars_17%>%
   group_by(SEX)%>%
   summarise(MeanAge = mean(AgeAtEnd),
             TotalEntries=sum(Entered),
             Caseload = sum(InAtEnd))
```

SEX	MeanAge	TotalEntries	Caseload
1	8.021688	138083	228589
2	8.179632	131649	214092

Grouping and summarizing: by state

```
table1<-afcars_17%>%
   group_by(St)%>%
   summarise(MeanAge = mean(AgeAtEnd),
             TotalEntries=sum(Entered),
             Caseload = sum(InAtEnd))
```

St	MeanAge	TotalEntries	Caseload
AK	7.417093	1328	2766
AL	8.177410	4095	5631
AR	7.167915	3778	4776
AZ	7.758218	10054	15028
CA	8.283262	28015	51867
CO	9.352652	5134	5704

Grouping and summarizing: by state and sex

```
table1<-afcars_17%>%
  group_by(St, SEX)%>%
  summarise(MeanAge = mean(AgeAtEnd),
            TotalEntries=sum(Entered),
            Caseload = sum(InAtEnd))
```

St	SEX	MeanAge	TotalEntries	Caseload
AK	1	7.415414	643	1375
AK	2	7.418787	685	1391
AL	1	7.747501	2070	2859
AL	2	8.625935	2025	2772
AR	1	6.961627	1907	2425
AR	2	7.379384	1871	2351

Moving from long to wide: Mean Age

```
wide1<-table1%>%
  select(St, SEX, MeanAge)%>%
  spread(key = SEX, value = MeanAge)
```

St	1	2
AK	7.415414	7.418787
AL	7.747501	8.625935
AR	6.961627	7.379384
AZ	7.832072	7.679496
CA	8.146840	8.424892
CO	9.984347	8.502218

Convert full data to wide by sex

```
wide_age<-table1%>%
  select(St, SEX, MeanAge)%>%
  spread(key = SEX, value = MeanAge,
         sep = "age")
wide_caseload<-table1%>%
  select(St, SEX, Caseload)%>%
  spread(key = SEX, value = Caseload,
         sep = "caseload" )
wide_entries<-table1%>%
  select(St, SEX, TotalEntries)%>%
  spread(key = SEX, value = TotalEntries,
         sep = "entries")
```

How to merge them?

```
names(wide_age)
## [1] "St"        "SEXage1" "SEXage2"
names(wide_caseload)
## [1] "St"              "SEXcaseload1"
"SEXcaseload2"
names(wide_entries)
## [1] "St"             "SEXentries1"
"SEXentries2"
```

Use St as our key column

```
wide_merge<-left_join(
  wide_age,
  wide_caseload
)

## Joining, by = "St"

wide_merge<-left_join(
  wide_merge,
  wide_entries
)

## Joining, by = "St"
```

Adding population data

From seer.cancer.gov/popdata

```
pop<-read_fwf("us.1990_2017.singleages.adjusted.txt",
              fwf_widths(c(4, 2, 2, 3,
                           2, 1, 1, 1,
                           2, 8),
                         c("year", "state", "st_fips",
                           "cnty_fips", "reg", "race",
                           "hisp", "sex", "age",
"pop")))%>%
  mutate(age = as.numeric(age),
         pop = as.numeric(pop))%>%
  filter(age<21, year==2017)%>%
  rename(st = state)
```

Aggregate it to state, year, seX

```
pop_st<-pop%>%
  group_by(st, sex, age)%>%
  summarise(pop = sum(pop))
```

Let’s get per capita entry rates by age and sex

```
percap<-left_join(
  afcars_17%>%
    group_by(SEX, St, AgeAtEnd)%>%
    summarise(entries = sum(Entered))%>%
    rename(st = St, sex = SEX, age =
AgeAtEnd),
  pop_st)%>%
  filter(age<19)

## Joining, by = c("sex", "st", "age")
```

What did this make?

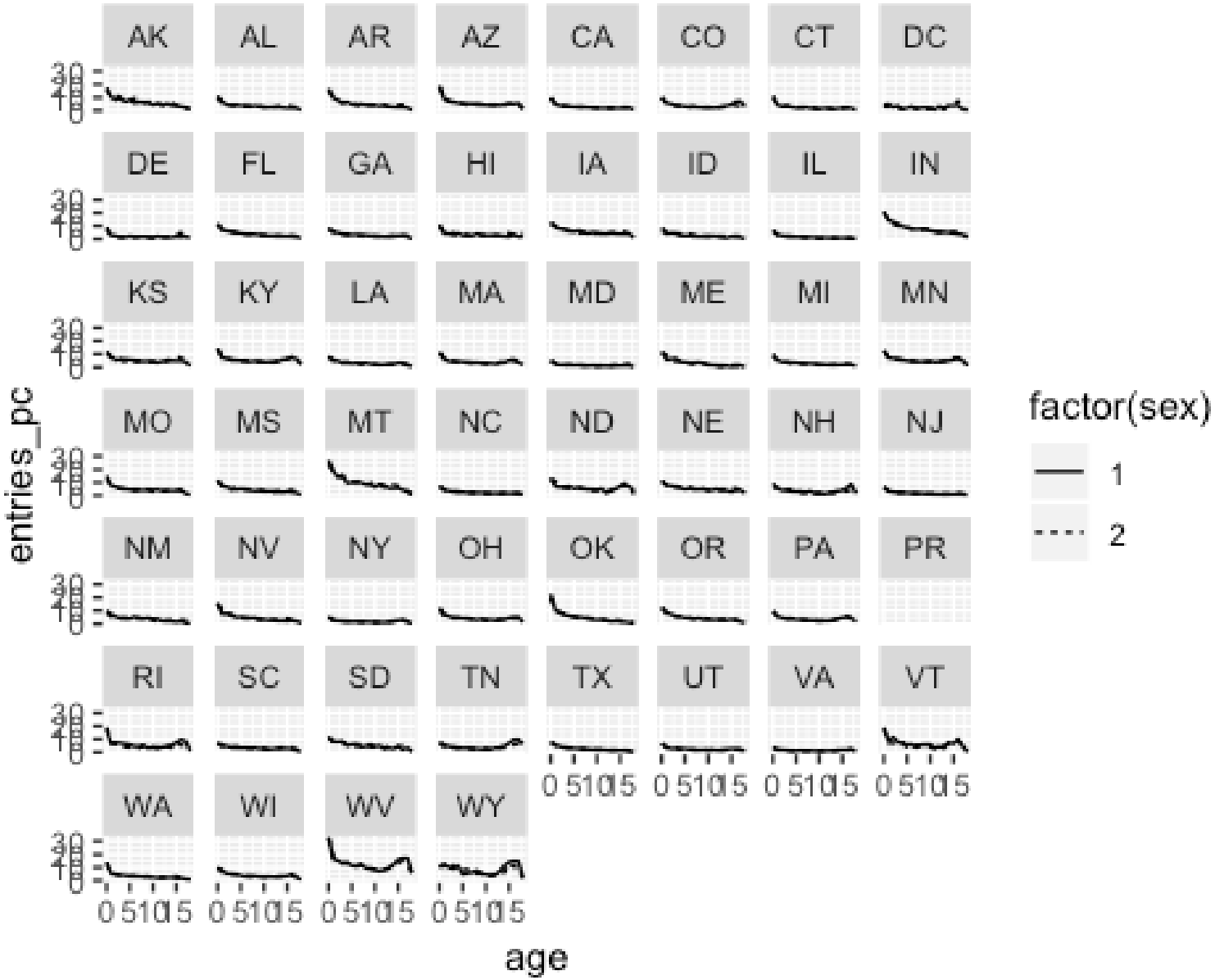
sex	st	age	entries	pop
1	AK	0	91	5673
1	AK	1	59	5595
1	AK	2	45	5369
1	AK	3	56	5433
1	AK	4	39	5555
1	AK	5	44	5344

Calculate per capita rates

```
percap<-percap%>%
  mutate(entries_pc = entries / pop * 1e3)

ggplot(percap,
       aes(x = age,
           y = entries_pc,
           lty = factor(sex))) +
  geom_line() +
  facet_wrap(~st)
```

Age of entry by sex for every state in the US



Draw a random sample

- Useful when working with large datasets to test models (e.g. NCANDS, AFCARS)
- Random subsets make code development more efficient, as computation on large datasets can take a LONG time

```
sample<-afcars%>%
  sample_frac(size = 0.1)
## Draw a 10 percent random sample
## Model placement stability for latest episode
Model_1<-glm(NumPlep ~ AgeAtEnd + Entered + Sex,
   data = sample, family = "poisson")
```

QUESTIONS?

FRANK EDWARDS
NDACAN STATISTIAN
ASSISTANT PROFESSOR, Rutgers
frank.edwards@rutgers.edu

Next week…

August 14, 2019

Linking NCANDS, AFCARS, and NYTD Data

Presenter: Michael Dineen