Stat 240 week 3: SQL 1

Dr. Dave Campbell
dac5@SFU.ca

Go on Exchange!

Apply by Jan 25: https://www.sfu.ca/students/studyabroad/exchanges.html

View spaces available: https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf

Looking at Exchange plot code with examples from last day

```
exchanges = read.csv("exchange2019.csv")

exchanges2 = read.csv("exchange2019.2.csv")

attach(exchanges2)

plot(table(Country),las=2,xlab="",ylab="Availability",main="Count of Schools & Terms Available per Country")

detach(exchanges2)

attach(exchanges)

count = apply(exchanges[,4:6],1,sum)

datause = aggregate(count, by=list(Country),FUN=sum,na.rm=T)

colnames(datause) = c("country","Count of # Schools and Terms Availble for Exchange \n Apply by Jan 25 Application Deadline")

library(rworldmap)

spdf = joinCountryData2Map(datause, joinCode="NAME", nameJoinColumn="country")

here = mapCountryData(spdf, nameColumnToPlot="Count of # Schools and Terms Availble for Exchange \n Apply by Jan 25 Application Deadline", catMethod="fixedWidth",numCats = 24,lwd=1)

do.call(addMapLegend, c(here,sigFigs=2,legendLabels="all",legendIntervals="page"))
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/pdf/2019_20%20Exchange%20Terms%20Available%20-%20Jan%2025%20Deadline.pdf into a plot

```
exchanges = read.csv("exchange2019.csv")

exchanges2 = read.csv("exchange2019.2.csv")

attach(exchanges2)

plot(table(Country),las=2,xlab="",ylab="Availability",main="Count of Schools & Terms Available per Country”)

detach(exchanges2)

attach(exchanges)

count = apply(exchanges[,4:6],1,sum)

datause = aggregate(count, by=list(Country),FUN=sum,na.rm=T)

colnames(datause) = c("country","Count of # Schools and Terms Availble for Exchange \n Apply by Jan 25 Application Deadline")

library(rworldmap)

spdf = joinCountryData2Map(datause, joinCode="NAME", nameJoinColumn="country")

here = mapCountryData(spdf, nameColumnToPlot="Count of # Schools and Terms Availble for Exchange \n Apply by Jan 25 Application Deadline", catMethod="fixedWidth",numCats = 24,lwd=1)

do.call(addMapLegend, c(here,sigFigs=2,legendLabels="all",legendIntervals="page"))
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/pdf/2019_20%20Exchange%20Terms%20Available%20-%20Jan%2025%20Deadline.pdf into a plot

```
exchanges = read.csv("exchange2019.csv")

exchanges2 = read.csv("exchange2019.2.csv")

attach(exchanges2)
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
exchanges = read.csv("exchanges.csv")

head(exchanges,10)# first 10 rows

tail(exchanges) # new function, prints last few rows

summary(exchanges)

#Is exchanges data.frame  or matrix?

summary(exchanges2)
```

Data Frame vs Matrix

	Data Frame	Matrix
Building	data.frame()	matrix(…)
Contents	numbers, factors, non-numbers	just numbers
Math	Might not let you do what you want to do	let's you do Math 232 or Math 240 operations
Converting	as.data.frame()	as.matrix()
Extraction	matrixname[,2] matrixname[,"columnname"] matrixname$columnname attach(matrixname);columnname	matrixname[,2] matrixname[,"columnname"]

from last time

```
allgrades[,"final"]

allgrades[,"final"]>0

allgrades[allgrades[,"final"]>0,"final"]

allgrades[,"Attend"]<4

allgrades[,"Attend"]==4
```

from last time

```
allgrades.dat.frm = as.data.frame(allgrades)
```

```
allgrades.dat.frm$final
```

```
allgrades.dat.frm$final>0
```

```
allgrades.dat.frm[allgrades.dat.frm$final>0,"final"]
```

```
allgrades.dat.frm$Attend<4
```

```
allgrades.dat.frm$Attend==4
```

Another equivalent

allgrades.dat.frm = as.data.frame(allgrades)

attach(allgrades.dat.frm)

final

final>0

final[final>0]

Attend<4

Attend == 4

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
exchanges2 = read.csv("exchange2019.2.csv")

attach(exchanges2)

plot(table(Country),    las=2,    xlab="",
ylab="Availability",
main="Count of Schools & Terms Available per Country")



table(Country) # Tally the number of occurrences
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
exchanges2 = read.csv("exchange2019.2.csv")

attach(exchanges2)

plot(table(Country),    las=2,    xlab="",
ylab="Availability",
main="Count of Schools & Terms Available per Country")

detach(exchanges2)
```

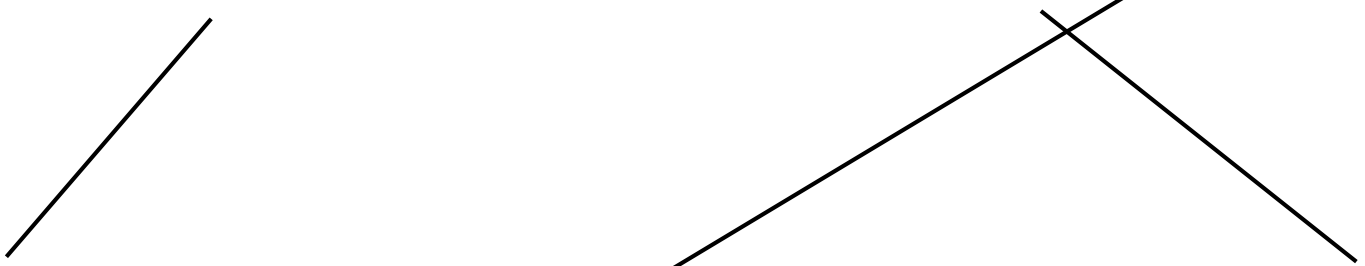
Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/pdf/2019_20%20Exchange%20Terms%20Available%20-%20Jan%2025%20Deadline.pdf into a plot

```
exchanges = read.csv("exchange2019.csv")

exchanges2 = read.csv("exchange2019.2.csv")

attach(exchanges2)

plot(table(Country),las=2,xlab="",ylab="Availability",main="Count of Schools & Terms Available per Country”)

detach(exchanges2)

attach(exchanges)

count = apply(exchanges[,4:6],1,sum)

datause = aggregate(count, by=list(Country),FUN=sum,na.rm=T)

colnames(datause) = c("country","Count of # Schools and Terms Availble for Exchange \n Apply by Jan 25 Application Deadline")

library(rworldmap)

spdf = joinCountryData2Map(datause, joinCode="NAME", nameJoinColumn="country")

here = mapCountryData(spdf, nameColumnToPlot="Count of # Schools and Terms Availble for Exchange \n Apply by Jan 25 Application Deadline", catMethod="fixedWidth",numCats = 24,lwd=1)

do.call(addMapLegend, c(here,sigFigs=2,legendLabels="all",legendIntervals="page"))
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

attach(exchanges)

count = **apply**(exchanges[,4:6]**,1,sum**)



Apply the function **sum** along **dimension #1** of the data.frame (or matrix)

Result has the same rows as exchanges

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
attach(exchanges)

count = apply(exchanges[,4:6],1,sum)

datause = aggregate(count,
by=list(Country),FUN=sum,na.rm=T)
```

aggregate how?

do what kind of summary?

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

exchanges = read.csv("exchanges.csv")

attach(exchanges)

datause = **aggregate(count, by=list(Country),FUN=sum,na.rm=T)**

na.rm = remove **NA** values…

```
temp1 = c(1,2,3,4,5)

temp2 = c(1,2,3,4,NA)

sum(temp1)

sum(temp2)
```

```
temp1 = c(1,2,3,4,5)

temp2 = c(1,2,3,4,NA)

sum(temp1)

sum(temp2,na.rm=T)
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
exchanges = read.csv("exchanges.csv")

attach(exchanges)

datause = aggregate(count,
by=list(Country),FUN=sum,na.rm=T)

colnames(datause) = c("country","Count of #
Schools and Terms Availble for Exchange \n Apply
by Jan 25 Application Deadline")
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
exchanges = read.csv("exchange2019.csv")

exchanges2 = read.csv("exchange2019.2.csv")

attach(exchanges2)

plot(table(Country),las=2,xlab="",ylab="Availability",main="Count of Schools & Terms Available per Country")

detach(exchanges2)

attach(exchanges)

count = apply(exchanges[,4:6],1,sum)

datause = aggregate(count, by=list(Country),FUN=sum,na.rm=T)

colnames(datause) = c("country","Count of # Schools and Terms Availble for Exchange \n Apply by Jan 25 Application Deadline")

library(rworldmap)

spdf = joinCountryData2Map(datause, joinCode="NAME", nameJoinColumn="country")

here = mapCountryData(spdf, nameColumnToPlot="Count of # Schools and Terms Availble for Exchange \n Apply by Jan 25 Application Deadline", catMethod="fixedWidth",numCats = 24,lwd=1)

do.call(addMapLegend, c(here,sigFigs=2,legendLabels="all",legendIntervals="page"))
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
library(rworldmap)

spdf = joinCountryData2Map(datause, joinCode="NAME",
nameJoinColumn="country")

thisistheplot = mapCountryData(spdf, nameColumnToPlot="Count of
# Schools and Terms Availble for Exchange \n Apply by Jan 25
Application Deadline", catMethod="fixedWidth",numCats = 24,lwd=1)

do.call(addMapLegend, c(thisistheplot,sigFigs=2

                        ,legendLabels="all"

                        ,legendIntervals="page"))
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
library(rworldmap)

spdf = joinCountryData2Map(datause,
joinCode="NAME", nameJoinColumn="country")

#merging 2 datasets by matching “country” with
“name”

#More about this in the lab!
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
library(rworldmap)

spdf = joinCountryData2Map(datause,
joinCode="NAME", nameJoinColumn="country")

here = mapCountryData(spdf,
nameColumnToPlot="Count of # Schools and Terms
Availble for Exchange \n Apply by Jan 25 Application
Deadline", catMethod="fixedWidth",numCats =
24,lwd=1)

#This makes a plot but also lets you modify it
```

Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
library(rworldmap)

spdf = joinCountryData2Map(datause, joinCode="NAME",
nameJoinColumn="country")

here = mapCountryData(spdf, nameColumnToPlot="Count of #
Schools and Terms Availble for Exchange \n Apply by Jan 25
Application Deadline", catMethod="fixedWidth",numCats =
24,lwd=1)

do.call(addMapLegend,
c(here,sigFigs=2,legendLabels="all",legendIntervals="page"))

#This style of legend only works for this kind of map plot
```

SFU INTERNATIONAL SERVICES FOR STUDENTS

STUDY ABROAD
EXCHANGE & FIELD SCHOOLS SHORT-TERM SUMMER PROGRAMS

www.sfu.ca/students/studyabroad

- Earn credits toward degree while paying SFU tuition
- Exchange programs at over 120 institutions
- Field schools with SFU courses taught internationally

EXCHANGE DEADLINE: JAN 25, MAY 25
FIELD SCHOOL DEADLINE: VARIES

INTERNATIONAL STUDENT
ADVISING & PROGRAMS

www.sfu.ca/students/isap

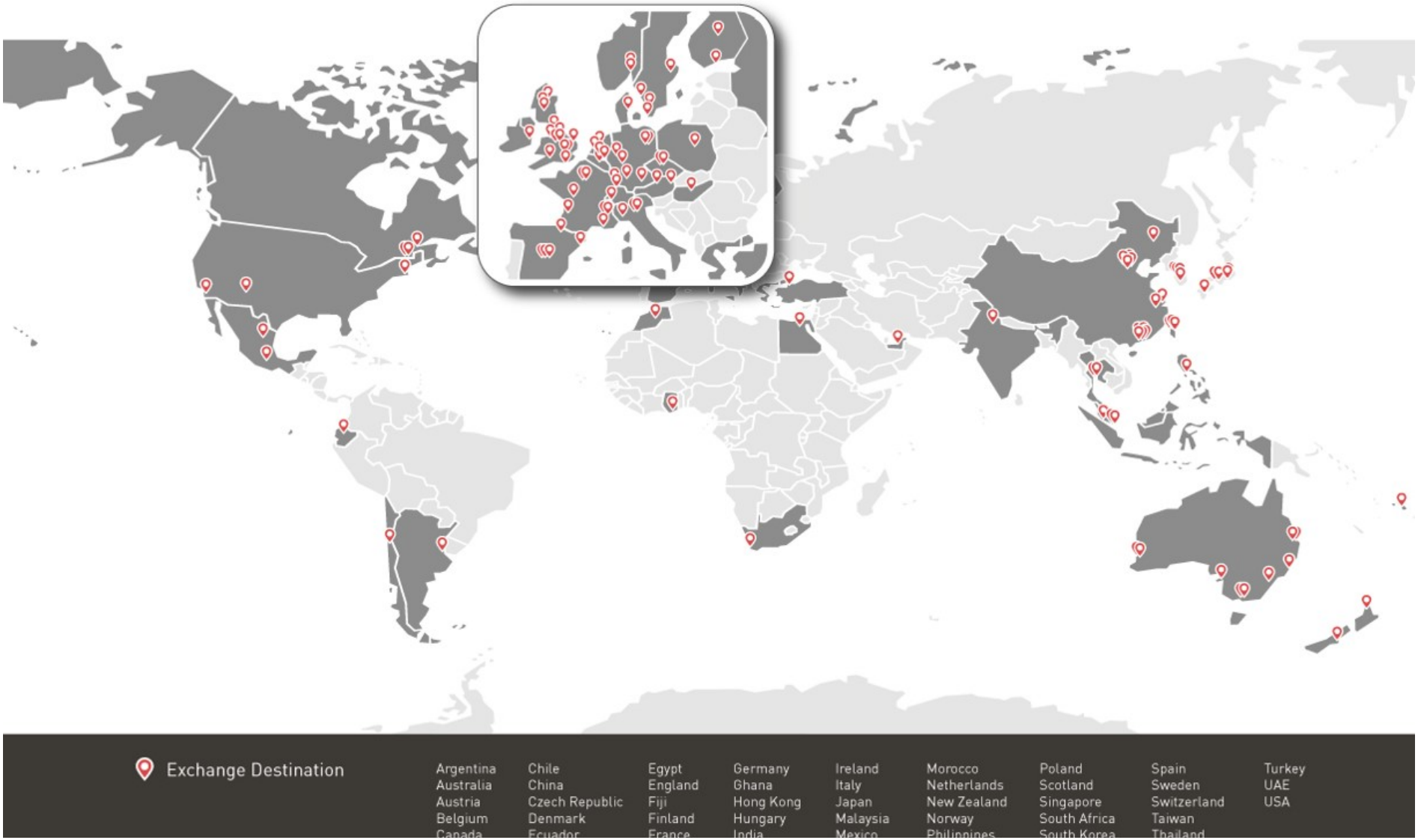
- Advising: Immigration, Medical Insurance, and Cultural Transition
- Workshops & events
- Student volunteer opportunities

INTERCULTURAL
ENGAGEMENT

www.sfu.ca/students/intercultural-engagement

- Global Student Centre - student lounge open to everyone!
- Global Community Mentorship Program
- Intercultural events and involvement opportunities

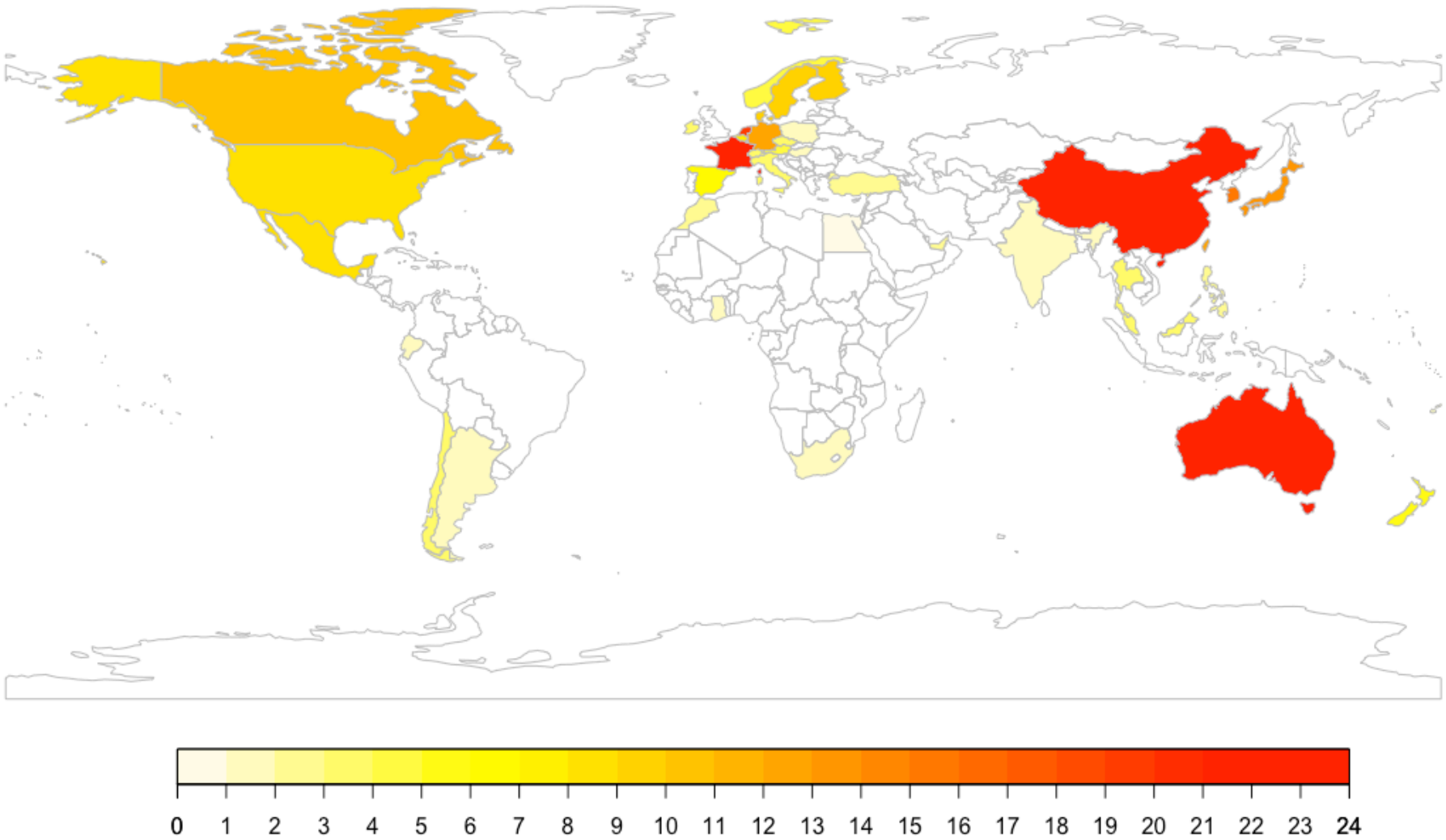
“In previous years we had posted the number of spaces, however we found that some students were misinterpreting the number of spaces. For example, a university with 10 spaces may be more competitive than another with only 1 or 2 spaces. Some students were not applying to the institutions with very low spaces, as they believed they would not get the space.”



Make this table https://www.sfu.ca/content/dam/sfu/students/studyabroad/exchange/2017_18%20Exchange%20Spaces%20-%20Jan%2025%20Deadline.pdf into a plot

```
attach(exchanges)

count = apply(exchanges[,4:6],1,sum)

datause = aggregate(count, by=list(Country),FUN=sum,na.rm=T)

colnames(datause) = c("country","Count of # Schools and Terms Availble for Exchange \n Apply by Jan 25 Application Deadline")

library(rworldmap)

spdf = joinCountryData2Map(datause, joinCode="NAME", nameJoinColumn="country")

here = mapCountryData(spdf, nameColumnToPlot="Count of # Schools and Terms Availble for Exchange \n Apply by Jan 25 Application Deadline", catMethod="fixedWidth",numCats = 24,lwd=1)

do.call(addMapLegend, c(here,sigFigs=2,legendLabels="all",legendIntervals="page"))
```

Count of # Schools and Terms Availble for Exchange
Apply by Jan 25 Application Deadline



install.packages

library

aggregate # do something to all values of ‘count’ a for a given value of Country

know the functions from class for exams.

****

About factors and levels

poke = read.csv("pokemon_2019.csv")

labels ≠ levels; labels = names, levels = factors

poke[,"Type_1"]

poke[,"Type_2"]

levels(poke[,"Type_1"])

levels(poke[,"Type_2"])

Note to Dave: start from line 247 to load data

Question 8 Lab 1 - did you do the right thing or just make something happen?

```
c(poke[,"Type_1"],poke[,"Type_2"])

# Basically, then is wrong:

factor(c(poke[,"Type_1"],poke[,"Type_2"]))
```

Note to Dave: start from line 247 to load data

Question 8: put everything into one big column / matrix

```
pokewithType1asType2 = poke

pokewithType1asType2[,"Type_2"]=poke[,"Type_1"]

^ This is because Type 1 is a subset of type 2 factor levels

stacked = rbind(poke,pokewithType1asType2)

# Now we need to make sure this worked.

levels(stacked[,"Type_2"]);

levels(poke[,"Type_1"]);

levels(poke[,"Type_2"])
```

Check you did it correctly, make the plots:

```
# Check a few pokemon:   look at Types 1 and 2 here:

poke[1:20,3:4]

stacked[1:20,"Type_2"]

stacked[801:820,"Type_2"]

sum(stacked[,"Type_2"]=="Bug");sum(poke[,"Type_1"]=="Bug")
+sum(poke[,"Type_2"]=="Bug")

par(mfrow=c(2,1));

boxplot(stacked[,"Attack"]~ stacked[,"Type_2"],main="Pokemon
Attacks",las=2,type='n',xlab="Pokemon Type",ylab="count")

boxplot(stacked[,"Defense"]~ stacked[,"Type_2"],main="Pokemon
Defense",las=2,type='n',xlab="Pokemon Type",ylab="count")
```

Chapter 7; Relational Database section

This process involved a lot of hacking data into the format that we needed. It was wasteful of memory (duplicate attack and defense info…)

The data format split pokemon type across two different columns of data. This made it challenging to extract information about pokemon types.

Data Table #1 Include pokemon names, attacks, defense,…
BUT NOT Types

All pokemon appear once.

Data Table #2, just include 2 columns pokemon and types.

A single pokemon; Bulbasaur will appear twice, once for GRASS and once for POISON.

This allows connections (relations) between tables to extract info more succinctly.

SQL - Text Chapter 7

Have a database but don't need all of it?

Have too much data for one computer?

Have another database? Want to merge them by matching column values?

Structured Query Language is the answer!

The Set Up

DataBase Management System (DBMS) holds the database - we'll connect to this

SQL is a language for sending requests to the database

DBMS performs the request and sends results out

R let's us analyze data

We'll do everything from within R

Libraries and start of SQL

This week we use the SQL database on your machine from within R - details are beyond this week.

Connecting to the database:

```
db_poke = dbConnect(RSQLite::SQLite(),
dbname="stat240Week3class.sqlite")

dbListTables(db_poke) # list tables in XXX
```

Libraries and start of SQL

About the database:

names(dbReadTable(db_poke, "Pokem"))

tail(dbReadTable(db_poke, "Pokem"))

ANYFUNCTION(dbReadTable(databasename, tablename))

Workflow

define a query using SQL syntax

retrieve query result using R

We'll define queries writing SQL code as a string of text, then send that to the database

quer = "SELECT * FROM Pokem WHERE Attack>100"

Apply **quer** to the database **db_poke**:
dbGetQuery(**db_poke**, **quer**)

Try to break it:

quer = "SeLeCt * frOm PokEm whERE AttACk>100"

dbGetQuery(db_poke, quer)

Selecting based on logical

```
quer = "SELECT Name FROM Pokem WHERE
isLegendary =='True'"

dbGetQuery(db_poke, quer)
```

Note the triple quote!

Composite logicals

```
quer = "SELECT Name FROM Pokem WHERE
isLegendary =='True' AND Generation ==3"

dbGetQuery(db_poke, quer)
```

This week and last week:

Goal is to plot the number of pokemon per generation:

quer = "SeLeCt Name, Generation FROM Pokem"

gens = dbGetQuery(db_poke, quer)

how do we use “aggregate” to get the number of pokemon per generation?

- aggregate(gens[,"Name"], by=list(gens[,"Generation"]),FUN=length)

This week and last week:

Goal is to plot the number of pokemon per generation:

quer = "SELECT Name, Generation FROM Pokem"

gens = dbGetQuery(db_poke, quer)

how do we use “aggregate” to get the number of pokemon per generation?

plot(…… type= ‘?’)

In lab today: Getting started with SQL, plus:

FROM, SELECT, JOIN

Hint:The textbook Chapter 7 is very very useful for this week and next.