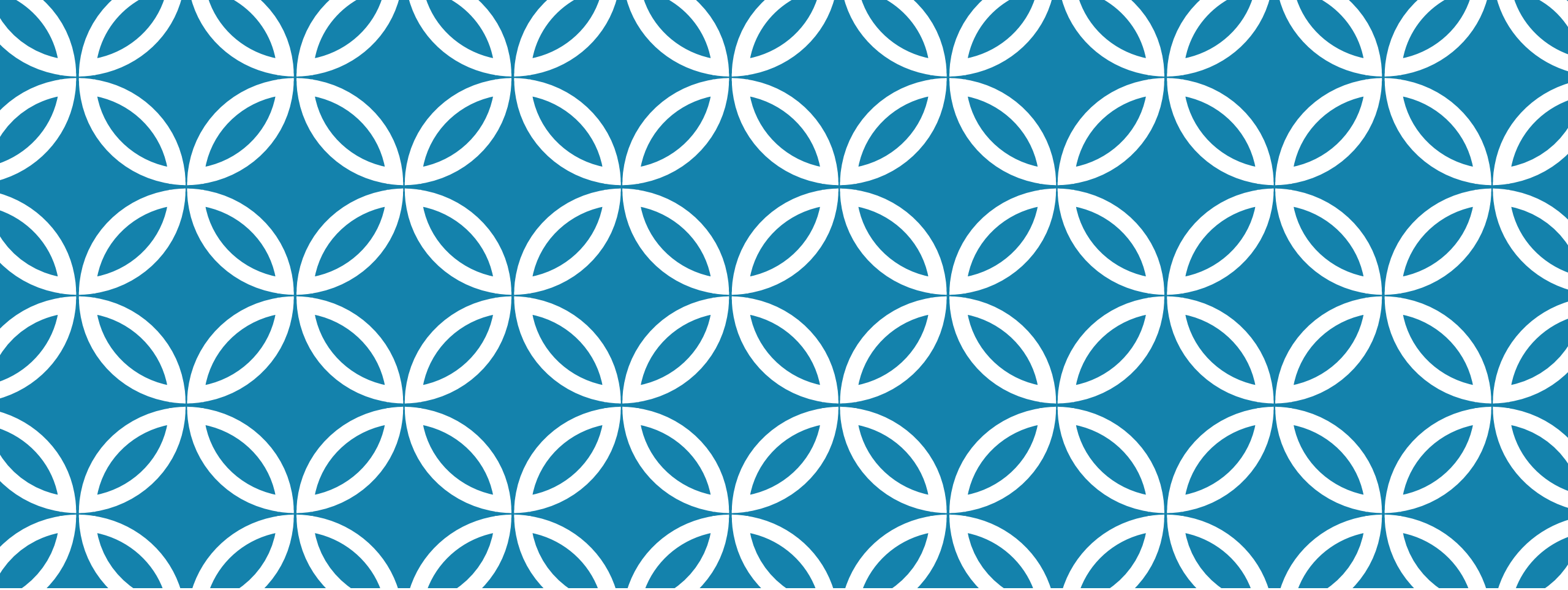


Reproducible Clinical Data Analysis with R and RStudio

Session 3

Exploring Data

April 2 and 3, 2018

Introduction to the Tidyverse

Stephan

Secure | https://www.datacamp.com/courses/introduction-to-the-tidyverse

Apps | Partners Apps | PPD | NHSN | Harvard Proxy | Sci-Hub | Add to Papers | IVIg calc | Path Hub | Other Bookmarks

DataCamp

Learn | My Groups | 10,990 XP

PAID COURSE

Introduction to the Tidyverse

Continue Course

4 hours | 16 Videos | 50 Exercises | 11,008 Participants | 4,150 XP | DOWNLOAD THE APP:

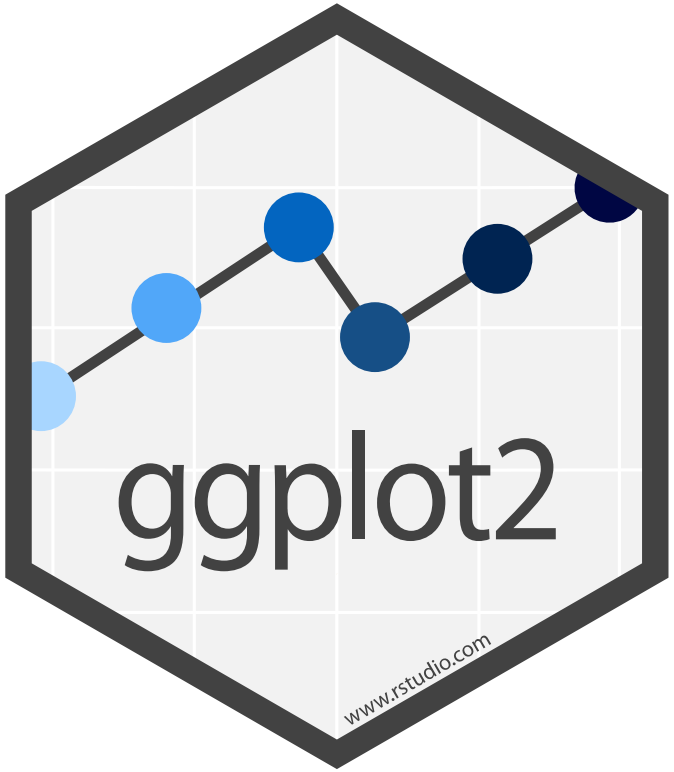
Course Description

Visualizing Data

Your Turn #1

How do you think the ESR results are distributed?

Discuss with your group.



A grammar of graphics for data visualization

```
library(tidyverse)
```

ggplot()

initialize a plot with ggplot()

data frame

+ sign (at end of line)

```
ggplot(data = esr) +
  geom_histogram(mapping = aes(x = Result))
```

type of graph

mappings inside aes() function

"aesthetic" mapping

To make **any** kind of graph:

1. Pick a "tidy" **data** frame

```
ggplot(data = <DATA>) +
  <GEOM_FUNCTION>(mapping = aes(<MAPPINGS>))
```

2. Choose a "**geom**" function

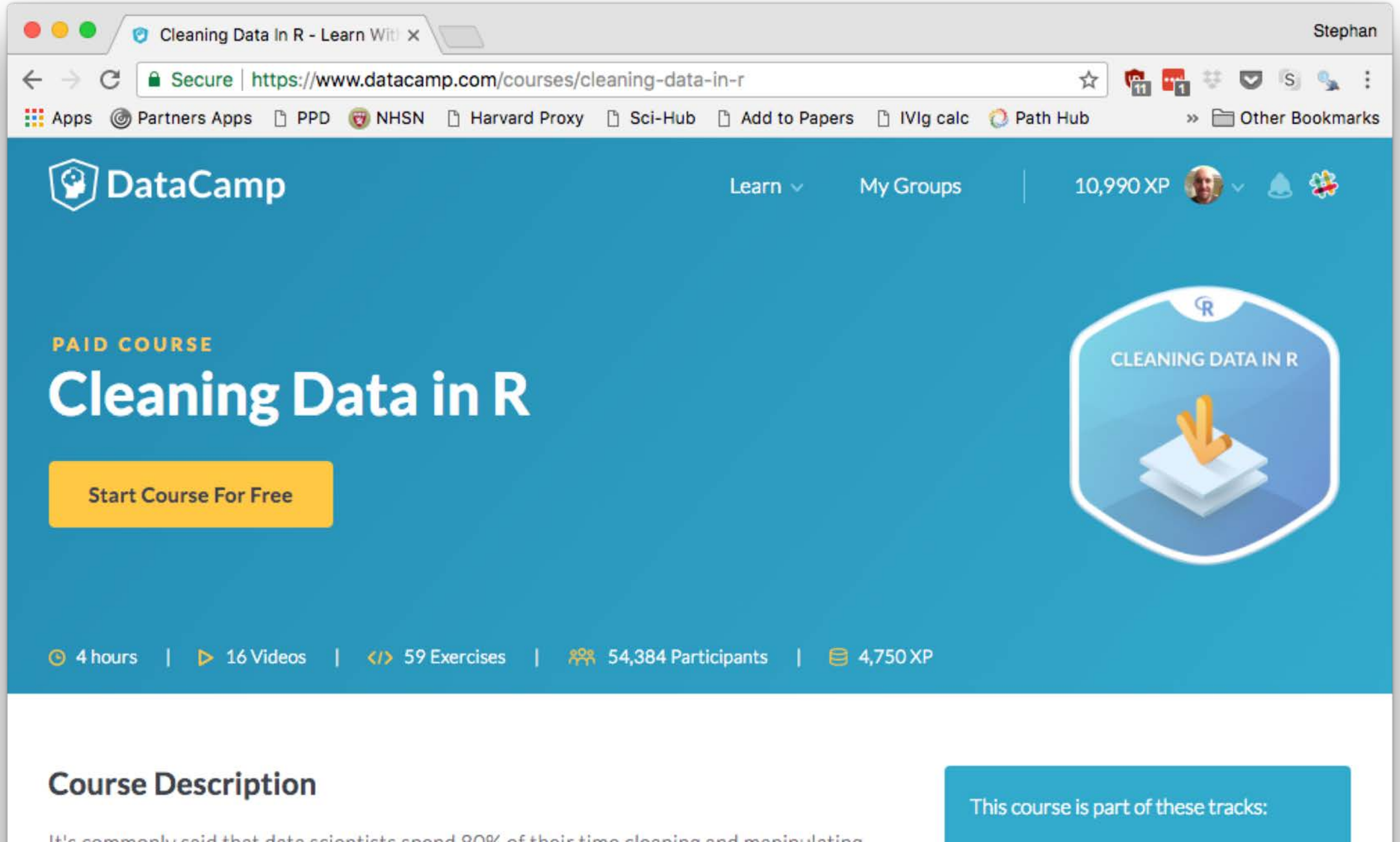
3. Write aesthetic **mappings**

1. Pick a "Tidy" Data Frame

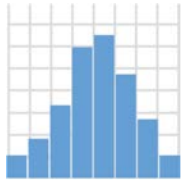
CollAge	PtNumber	PtSex	Result
7	5143567	M	22
42	3459254	F	5
19	2332467	F	5
80	3445732	M	89
41	7245673	F	12

A data set is **tidy** if:

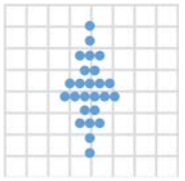
1. Each **variable** is in its own **column**
2. Each **case** is in its own **row**
3. Each **value** is in its own **cell**


Cleaning Data In R - Learn Wit
Stephan
Secure | https://www.datacamp.com/courses/cleaning-data-in-r
Apps
Partners Apps
PPD
NHSN
Harvard Proxy
Sci-Hub
Add to Papers
IVIg calc
Path Hub
Other Bookmarks
DataCamp
Learn
My Groups
10,990 XP
PAID COURSE
Cleaning Data in R
Start Course For Free
CLEANING DATA IN R
4 hours | 16 Videos | 59 Exercises | 54,384 Participants | 4,750 XP
Course Description
This course is part of these tracks:

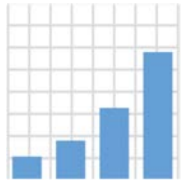
2. Choose a “Geom” Function



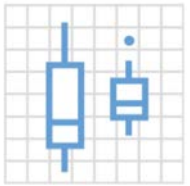
```
geom_histogram()
```



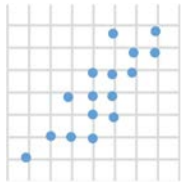
```
geom_dotplot()
```



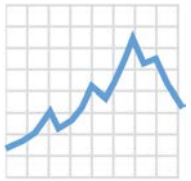
```
geom_bar()
```



```
geom_boxplot()
```



```
geom_point()
```

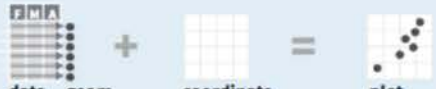


```
geom_line()
```

Data Visualization with ggplot2 : : CHEAT SHEET

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same components: a **data** set, a **coordinate system**, and geoms—visual marks that represent data points.



To display values, map variables in the data to visual properties of the geom (**aesthetics**) like **size**, **color**, and **x** and **y** locations.

Complete the template below to build a graph.

```
ggplot (data = <DATA>) +                                  required
<GEOM_FUNCTION> (mapping = aes( <MAPPINGS> ),
 stat = <STAT> , position = <POSITION> ) +                Not required, sensible defaults supplied
<COORDINATE_FUNCTION> +
<FACET_FUNCTION> +
<SCALE_FUNCTION> +
<THEME_FUNCTION>
```

ggplot(data = mpg, **aes**(x = cty, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

aesthetic mappings | data | geom

qplot(x = cty, y = hwy, data = mpg, geom = "point") Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

a <- ggplot(economics, aes(date, unemploy))
b <- ggplot(seals, aes(x = long, y = lat))

a + geom_blank()
(Useful for expanding limits)

b + geom_curve(aes(yend = lat + 1, xend=long+1,curvature=z)) - x, xend, y, yend, alpha, angle, color, curvature, linetype, size

a + geom_path(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, group, linetype, size

a + geom_polygon(aes(group = group))
x, y, alpha, color, fill, group, linetype, size

b + geom_rect(aes(xmin = long, ymin=lat, xmax= long + 1, ymax = lat + 1)) - xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

a + geom_ribbon(aes(ymin=unemploy - 900, ymax=unemploy + 900)) - x, ymax, ymin, alpha, color, fill, group, linetype, size

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

b + geom_abline(aes(intercept=0, slope=1))
b + geom_hline(aes(yintercept = lat))
b + geom_vline(aes(xintercept = long))

b + geom_segment(aes(yend=lat+1, xend=long+1))
b + geom_spoke(aes(angle = 1:1155, radius = 1))

ONE VARIABLE continuous

c <- ggplot(mpg, aes(hwy)); c2 <- ggplot(mpg)

c + geom_area(stat = "bin")
x, y, alpha, color, fill, linetype, size

c + geom_density(kernel = "gaussian")
x, y, alpha, color, fill, group, linetype, size, weight

c + geom_dotplot()
x, y, alpha, color, fill

c + geom_freqpoly() x, y, alpha, color, group,

TWO VARIABLES

continuous x , continuous y

e <- ggplot(mpg, aes(cty, hwy))

e + geom_label(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE) x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

e + geom_jitter(height = 2, width = 2)
x, y, alpha, color, fill, shape, size

e + geom_point(), x, y, alpha, color, fill, shape, size, stroke

e + geom_quantile(), x, y, alpha, color, group, linetype, size, weight

e + geom_rug(sides = "bl"), x, y, alpha, color, linetype, size

e + geom_smooth(method = lm), x, y, alpha, color, fill, group, linetype, size, weight

e + geom_text(aes(label = cty), nudge_x = 1, nudge_y = 1, check_overlap = TRUE), x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

discrete x , continuous y

f <- ggplot(mpg, aes(class, hwy))

f + geom_col(), x, y, alpha, color, fill, group, linetype, size

f + geom_boxplot(), x, y, lower, middle, upper, ymax, ymin, alpha, color, fill, group, linetype, shape, size, weight

f + geom_dotplot(binaxis = "y", stackdir = "center"), x, y, alpha, color, fill, group

f + geom_violin(scale = "area"), x, y, alpha, color, fill, group, linetype, size, weight

discrete x , discrete y

g <- ggplot(diamonds, aes(cut, color))

continuous bivariate distribution

h <- ggplot(diamonds, aes(carat, price))

h + geom_bin2d(binwidth = c(0.25, 500))
x, y, alpha, color, fill, linetype, size, weight

h + geom_density2d()
x, y, alpha, colour, group, linetype, size

h + geom_hex()
x, y, alpha, colour, fill, size

continuous function

i <- ggplot(economics, aes(date, unemploy))

i + geom_area()
x, y, alpha, color, fill, linetype, size

i + geom_line()
x, y, alpha, color, group, linetype, size

i + geom_step(direction = "hv")
x, y, alpha, color, group, linetype, size

visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
j <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

j + geom_crossbar(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, group, linetype, size

j + geom_errorbar(), x, ymax, ymin, alpha, color, group, linetype, size, width (also **geom_errorbarh()**)

j + geom_linerange()
x, ymin, ymax, alpha, color, group, linetype, size

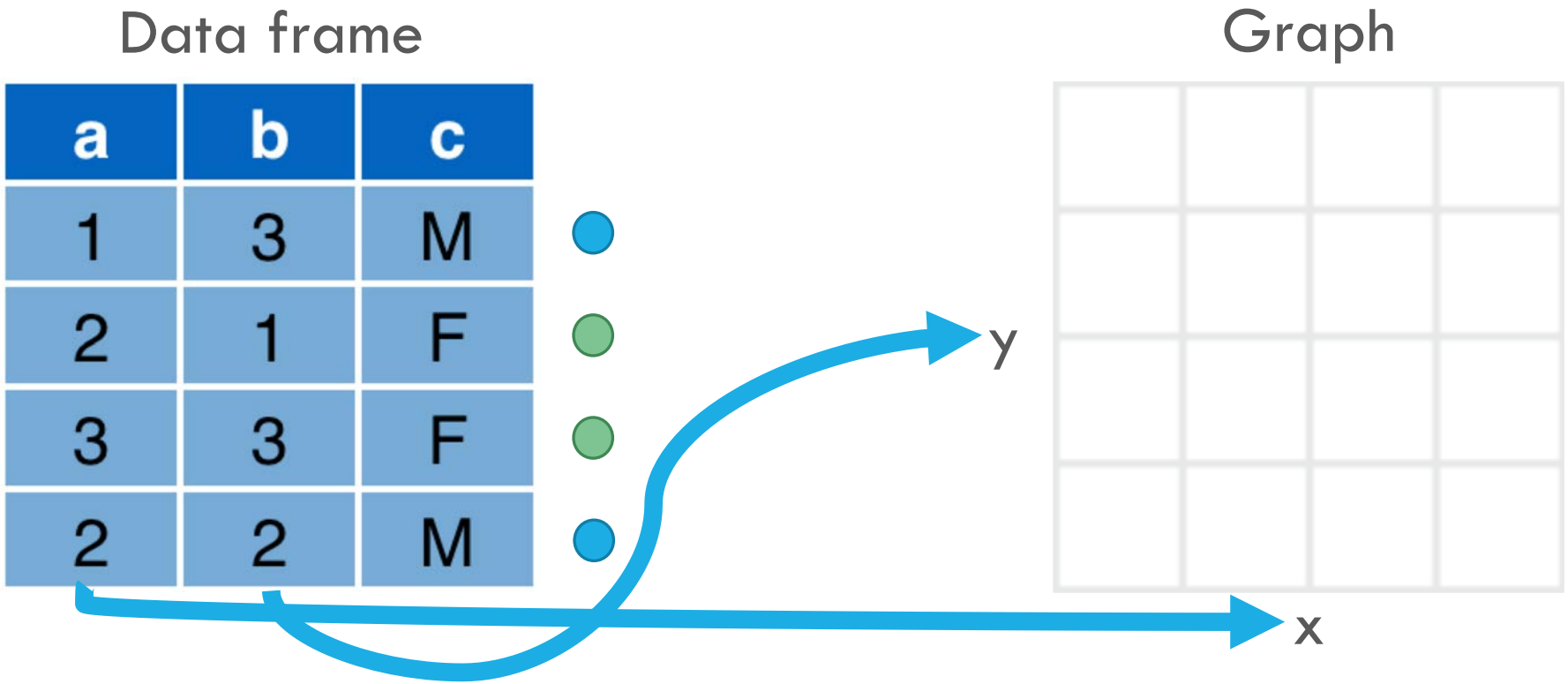
j + geom_pointrange()
x, y, ymin, ymax, alpha, color, fill, group, linetype, shape, size

maps

data <- data.frame(murder = USArrests$Murder, state = tolower(rownames(USArrests)))
map <- map_data("state")

3. Write Aesthetic Mappings

```
aes(x = a, y = b, color = c)
```



Your Turn #2

Download the files I sent to you by e-mail:

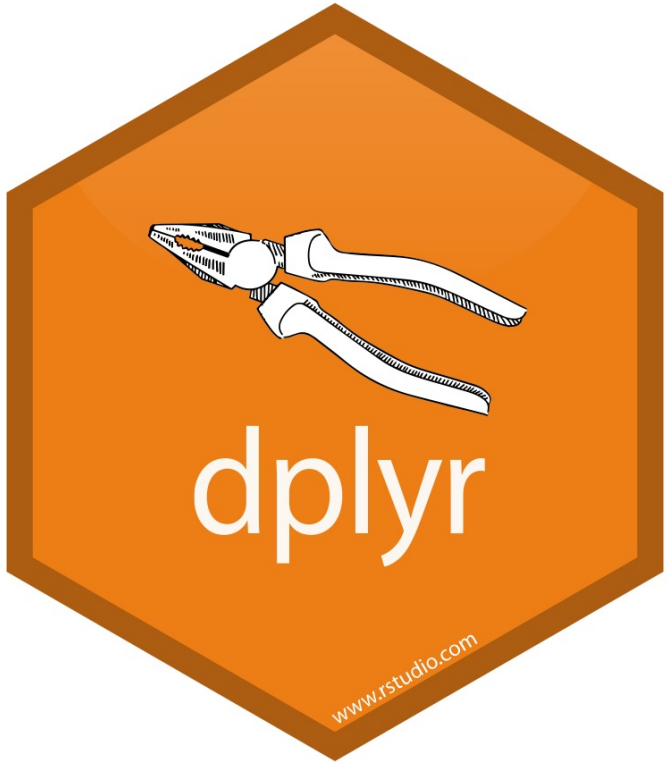
1. `03-exploring-data.Rmd`
2. `esr_data.csv`

Open `03-exploring-data.Rmd` in RStudio.

Complete the section "**Your Turn #2: Visualizing Data**".

Then, complete the section "**Visualizing Data**" on the handout.

Isolating Data



A grammar to transform rectangular data

```
library(tidyverse)
```

select()

Extract columns by name.

```
select(<DATA>, ...)
```

data frame

names of columns to extract

select()

Extract columns by name.

```
select(esr, CollAge, Result)
```

CollAge	PtNumber	PtSex	Result
7	5143567	M	22
42	3459254	F	5
19	2332467	F	5
80	3445732	M	89
41	7245673	F	12

CollAge	Result
7	22
42	5
19	5
80	89
41	12

filter()

Extract rows that meet logical criteria.

```
filter(<DATA>, <CONDITION>)
```

data frame

logical test
(return each row for which the test is TRUE)

filter()

Extract rows that meet logical criteria.

= sets
== compares

```
filter(esr, PtSex == "F")
```

CollAge	PtNumber	PtSex	Result
7	5143567	M	22
42	3459254	F	5
19	2332467	F	5
80	3445732	M	89
41	7245673	F	12

CollAge	PtNumber	PtSex	Result
42	3459254	F	5
19	2332467	F	5
41	7245673	F	12

Logical Tests

`x < y`	less than
`x > y`	greater than
`x == y`	equal to
`x <= y`	less than or equal to
`x >= y`	greater than or equal to
`x != y`	not equal to
`is.na(x)`	a missing value

arrange()

Order rows from smallest to largest.

```
arrange(<DATA>, ...)
```

data frame

names of columns to order by
(additional columns = tie breakers)

arrange()

Order rows from smallest to largest

descending order:
desc(CollAge)

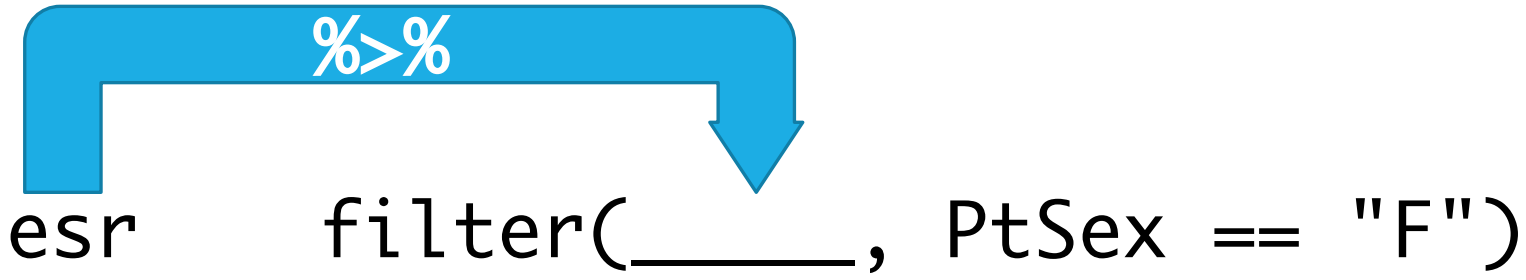
```
arrange(esr, CollAge)
```

CollAge	PtNumber	PtSex	Result
7	5143567	M	22
42	3459254	F	5
19	2332467	F	5
80	3445732	M	89
41	7245673	F	12

CollAge	PtNumber	PtSex	Result
7	5143567	M	22
19	2332467	F	5
41	7245673	F	12
42	3459254	F	5
80	3445732	M	89

%>%

The Pipe Operator %>%



Passes the object on the left as first argument to the function on the right.

```
filter(esr, PtSex == "F")
esr %>% filter(PtSex == "F")
```

”then”

```
esr %>%
  select(CollAge, PtSex, Result) %>%
  filter(PtSex == "F") %>%
  arrange(Result)
```

Your Turn #3

Return to `03-exploring-data.Rmd`.

Complete the section "**Your Turn #3: Isolating Data**".

Then, complete the section "**Isolating Data**" on the handout.

Augmenting Data

mutate()

Create new columns.

```
mutate(<DATA>, ...)
```

data frame

name-value expressions
`column_name = value`

mutate()

Create new columns.

```
mutate(esr,
       BirthYr = 2017 - CollAge)
```

CollAge	PtNumber	PtSex	Result
7	5143567	M	22
42	3459254	F	5
19	2332467	F	5
80	3445732	M	89
41	7245673	F	12

CollAge	PtNumber	PtSex	Result	BirthYr
7	5143567	M	22	2010
42	3459254	F	5	1975
19	2332467	F	5	1998
80	3445732	M	89	1937
41	7245673	F	12	1976

Join Data in R | DataCamp

Stephan

Secure | https://www.datacamp.com/courses/joining-data-in-r-with-dplyr

Apps | Partners Apps | PPD | NHSN | Harvard Proxy | Sci-Hub | Add to Papers | IVIg calc | Path Hub | Other Bookmarks

DataCamp

Learn | My Groups | 10,990 XP

PAID COURSE

Joining Data in R with dplyr

Continue Course

4 hours | 20 Videos | 84 Exercises | 15,546 Participants | 6,550 XP

Course Description

This course builds on what you learned in Data Manipulation in R with dplyr by showing you

This course is part of these tracks:

Grouping and Summarizing Data

```
esr %>%
    group_by(PtSex) %>%
    summarize(MeanAge = mean(CollAge),
              MeanESR = mean(Result))
```

CollAge	PtNumber	PtSex	Result
7	5143567	M	22
42	3459254	F	5
19	2332467	F	5
80	3445732	M	89
41	7245673	F	12

CollAge	PtNumber	PtSex	Result
42	3459254	F	5
19	2332467	F	5
41	7245673	F	12

7	5143567	M	22
80	3445732	M	89

PtSex	MeanAge	MeanESR
F	34	11
M	43.5	55.6

Summary (Aggregate) Functions

Set argument `na.rm = TRUE` to ignore missing values (NA)

`n()`	total count
`n_distinct(x)`	count of distinct values
`mean(x)`	mean
`median(x)`	median
`sum(x)`	sum
`sd(x)`	standard deviation

Data Transformation with dplyr : : CHEAT SHEET

dplyr functions work with pipes and expect **tidy data**. In tidy data:

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

&

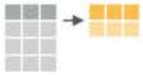
Each **observation**, or **case**, is in its own **row**

pipes

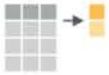
x %>% f(y) becomes **f(x, y)**

Summarise Cases

These apply **summary functions** to columns to create a new table. Summary functions take vectors as input and return one value (see back).



summarise(.data, …)
Compute table of summaries. Also **summarise_**().
summarise(mtcars, avg = mean(mpg))



count(x, ..., wt = NULL, sort = FALSE)
Count number of rows in each group defined by the variables in … Also **tally**().
count(iris, Species)

VARIATIONS

summarise_all() - Apply funs to every column.
summarise_at() - Apply funs to specific columns.
summarise_if() - Apply funs to all cols of one type.

Group Cases

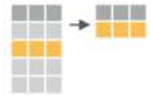
Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

mtcars %>%

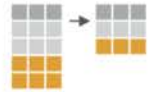
Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table. Use a variant that ends in _ for non-standard evaluation friendly code.



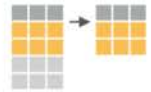
filter(.data, …**)** Extract rows that meet logical criteria. Also **filter_()**. *filter(iris, Sepal.Length > 7)*



distinct(.data, ..., .keep_all = FALSE**)** Remove rows with duplicate values. Also **distinct_()**. *distinct(iris, Species)*

sample_frac(tbl, size = 1, replace = FALSE, weight = NULL, .env = parent.frame()**)** Randomly select fraction of rows.
sample_frac(iris, 0.5, replace = TRUE)

sample_n(tbl, size, replace = FALSE, weight = NULL, .env = parent.frame()**)** Randomly select size rows. *sample_n(iris, 10, replace = TRUE)*



slice(.data, …**)** Select rows by position. Also **slice_()**. *slice(iris, 10:15)*

top_n(x, n, wt**)** Select and order top n entries (by group if grouped data). *top_n(iris, 5, Sepal.Width)*

Logical and boolean operators to use with filter()

<	<=	is.na()	%in%	|	xor()
>	>=	!is.na()	!	&	

See **?base::logic** and **?Comparison** for help.

ARRANGE CASES

arrange(.data, …**)** Order rows by values of a column or columns (low to high), use with

Column functions return a set of columns as a new table. Use a variant that ends in _ for non-standard evaluation friendly code.

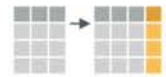
select(.data, …**)**
Extract columns by name. Also **select_if()**
select(iris, Sepal.Length, Species)

Use these helpers with select (),
e.g. select(iris, starts_with("Sepal"))

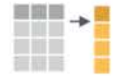
contains(match**)**	**num_range(**prefix, range**)**	**:**, e.g. mpg:cyl
ends_with(match**)**	**one_of(**…**)**	**-**, e.g, -Species
matches(match**)**	**starts_with(**match**)**	

MAKE NEW VARIABLES

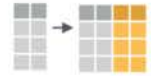
These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).



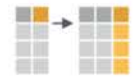
mutate(.data, …**)**
Compute new column(s).
mutate(mtcars, gpm = 1/mpg)



transmute(.data, …**)**
Compute new column(s), drop others.
transmute(mtcars, gpm = 1/mpg)



mutate_all(.tbl, .funs, …**)** Apply funs to every column. Use with **funs()**.
mutate_all(faithful, funs(log(.), log2(.)))



mutate_at(.tbl, .cols, .funs, …**)** Apply funs to specific columns. Use with **funs()**, **vars()** and the helper functions for select().

Your Turn #4

Return to `03-exploring-data.Rmd`.

Complete the section "**Your Turn #4: Grouping and Summarizing Data**".

Then, complete the section "**Grouping and Summarizing Data**" on the handout.