

Introduction to DAV 5300

Computational Mathematics and Statistics

Jason Bryer, Ph.D.

January 20, 2026

Agenda

- Introductions
- Syllabus
- Class meetups
- Course Schedule
- Assignments (how you will be graded)
- Software setup
- Brief introduction to R

While waiting, please complete this formative assessment:



A little about me...

- Earned my Ph.D. in Educational Psychology and Methodology from the University at Albany.
Dissertation: [A National Study Comparing Charter and Traditional Public Schools Using Propensity Score Analysis](#)
- Assistant Professor at CUNY in Data Science and Information Systems
- Principal Investigator for a Department of Education Grant to develop and test the Diagnostic Assessment and Achievement of College Skills (www.DAACS.net)
- Authored over a dozen R packages including:
 - [likert](#)
 - [clav](#)
 - [VisualStats](#)
- Specialize in propensity score methods. Three new methods/R packages developed include:
 - [multilevelPSA](#)
 - [TriMatch](#)
 - [PSAboot](#)

Also a Father...



Runner...



And photographer.



Your turn...

What is your name?

What are your career goals?

Something unique about you (e.g. hobby, interest, favorite movie)?

Syllabus



Syllabus and course materials are here:

<https://spring2026.dav5300.net>

We will use Canvas primary for submitting assignments only. Please submit PDFs.

PDFs are preferred for the homework as there is some LaTeX formatting in the R markdown files. The `tinytex` R package helps with install LaTeX, but you can also install LaTeX using [MiKTeX](#) (for Windows) and [BasicTeX](#) (for Mac).



Class Meetings

Class will meet every Tuesday.

In order to get the most out of this class attendance is required.

One Minute Papers - Complete the one minute paper after each Meetup (whether you watch live or watch the recordings). It should take approximately one to two minutes to complete.

Schedule

Topic	Date
Intro to the Course / Intro to Data	2026-01-20
Summarizing Data	2026-01-27
Probability and Distributions	2026-02-03
Foundation for Inference	2026-02-10
NO CLASS	2026-02-17
Inference for Categorical Data	2026-02-24
NO CLASS - Purim-No	2026-03-03
Inference for Numerical Data	2026-03-10
Linear Regression	2026-03-17
Maximum Likelihood Estimation and Logistic Regression	2026-03-24
Multiple Regression	2026-03-31
NO CLASS - Spring Break	2026-04-07
Conferences (online)	2026-04-14
Predictive Modeling and CART Methods	2026-04-21
Bayesian Analysis and Propensity Score Analysis	2026-04-28
Presentations	2026-05-05
Final Exam	2026-05-12

Assignments are due before the next class.

Textbooks

OpenIntro Statistics by David Diaz, Mine Çetinkaya-Rundel, and Christopher D Barr.

Learning Statistics with R by Danielle Navaro - We will only use the Bayesian chapter from this book.

Optional

R for Data Science by Hadley Wickham and Garrett Grolemund - Recommended reference for those new to R.

Assignments

Labs (30%) - Labs are designed to provide you an opportunity to apply statistical concepts using statistical software.

Textbook questions (15%) - The assigned questions from the textbook provide an opportunity to assess conceptional understandings.

Participation (10%) - You are expected to attend every class and to complete a **one minute paper** at the conclusion of class.

Data Project (25%) - In a group of 2 to 3 students will present the results of analysis using a data set of your choice. More details will be provided a few weeks into the class.

Final exam (20%) - A multiple choice exam will be given on the last day of class.

All assignments are due on Tuesday. Assignments submitted late will be penalized. Assignments will not be accepted more than two weeks after their due date.

Academic Integrity

With the exception of the data project, I expect you to complete all assignments (e.g. homework, labs) on your own. It is fine to ask questions of your peers and professor, but working together and/or sharing answers is not allowed.

Yeshiva's Policy

The submission by a student of any examination, course assignment, or degree requirement is assumed to guarantee that the thoughts and expressions therein not expressly credited to another are literally the student's own. Evidence to the contrary will result in appropriate penalties. For more information, visit <https://www.yu.edu/academic-integrity>.

Use of Artificial Intelligence (AI)

First, AI is a marketing term. I prefer to be more specific regarding what we are doing:

1. Machine Learning (ML) - This course will provide the foundations for how ML algorithms work. Generally speaking, the goal is to predict some known (and sometimes unknown in the case of unsupervised learning models) outcome.
2. Large Language Models (LLM) - This is often what people mean when they say AI. This includes products like ChatGPT, Anthropic, Google Gemini, etc. LLMs generate text, images, videos, etc. from a prompt.

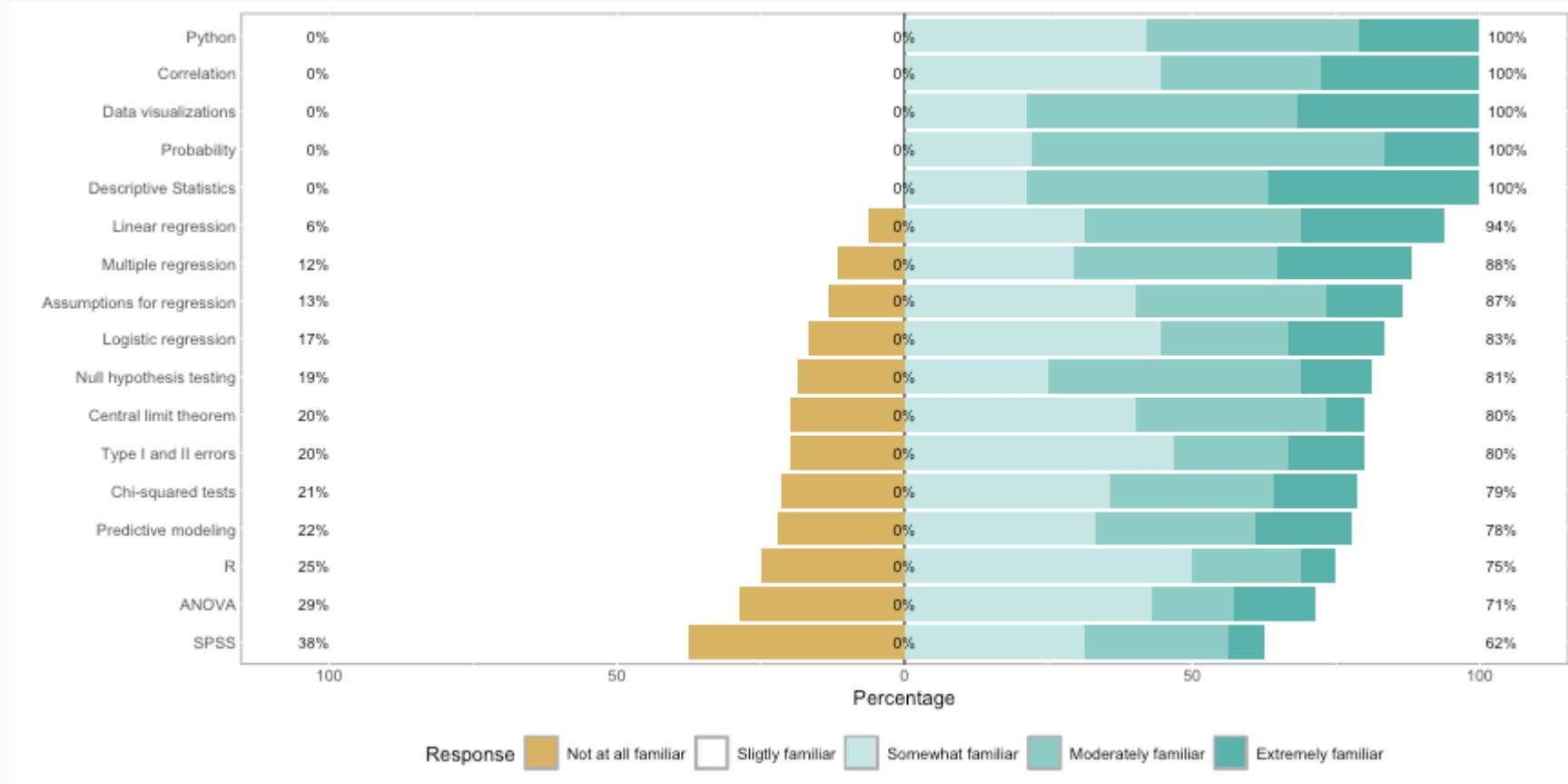
The goal of this course is for *you to develop the foundation knowledge and skills to do statistics*. Using chat bots to do the assignments subverts this goal. **The content generated by LLMs is often wrong!** If you use LLMs to assist in completing the assignments, you must include the prompt and response in your submission.

Communication

- Email: jason.bryer@yu.edu.
- Canvas
- Office hours before and after class and by appointment.

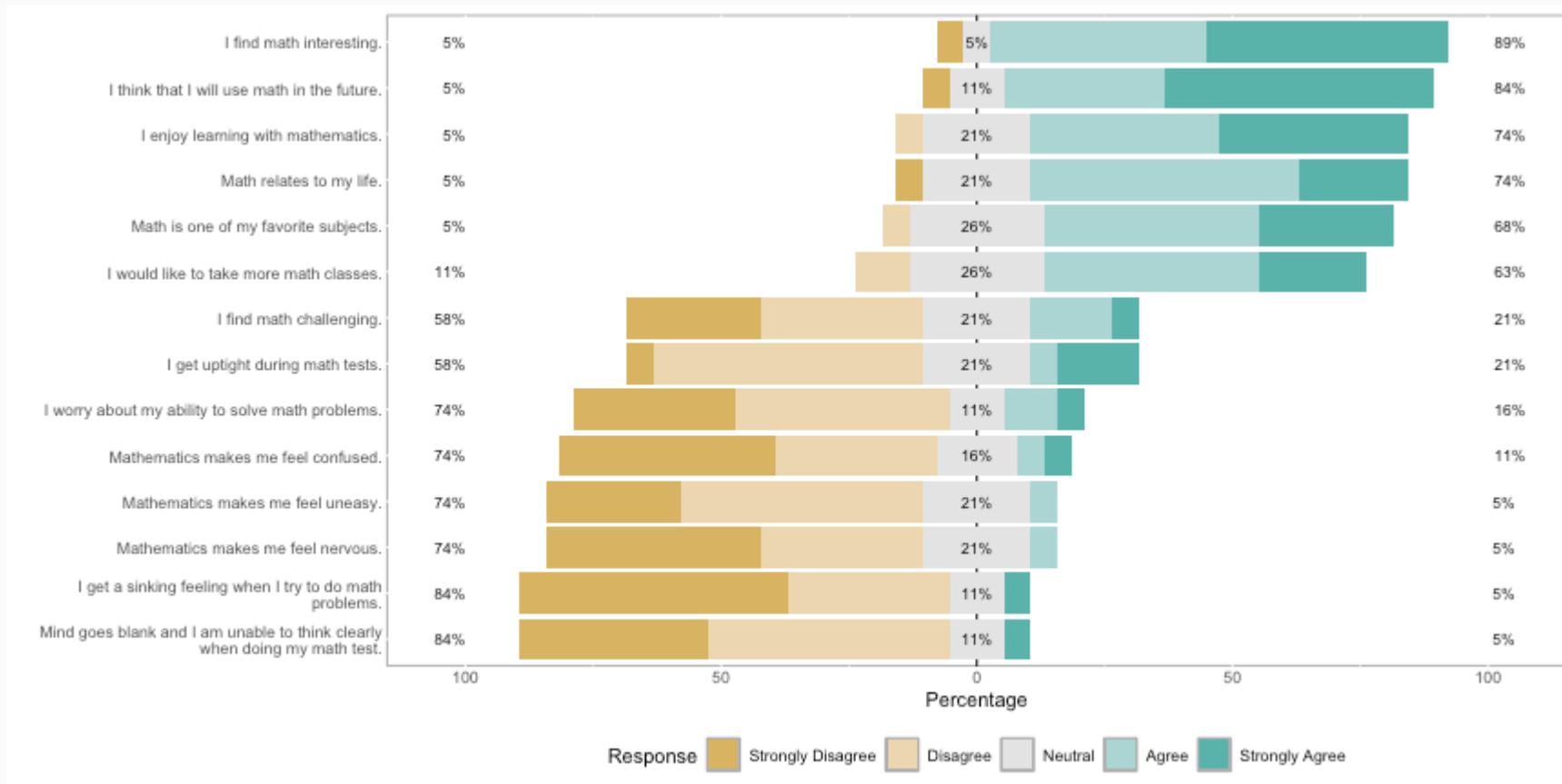
Familiarity with Statistical Topics

```
likert(stats.results) |> plot(center = 2.5)
```



Math Anxiety Survey Scale

```
likert(mass.results) |> plot()
```



Software Setup

Why R?

There are many languages data scientists use. **R** is specifically designed for statistics. We will leverage many R packages that are specifically designed to conduct, teach, and communicate statistical analysis.

To be a well rounded data scientists, I believe you need to have experience in both R and Python. For this course:

- Use R for the labs (they are designed to help you learn the core commands).
- You may use Python or R for the homework and data project.



Software



This is an applied statistics course so we will make extensive use of the **R statistical programming language**.

- Install **R** and **RStudio** on your own computer. I encourage everyone to do this at some point by the end of the semester.

You will also need to have **LaTeX** installed as well in order to create PDFs. The **tinytex** R package helps with this process:

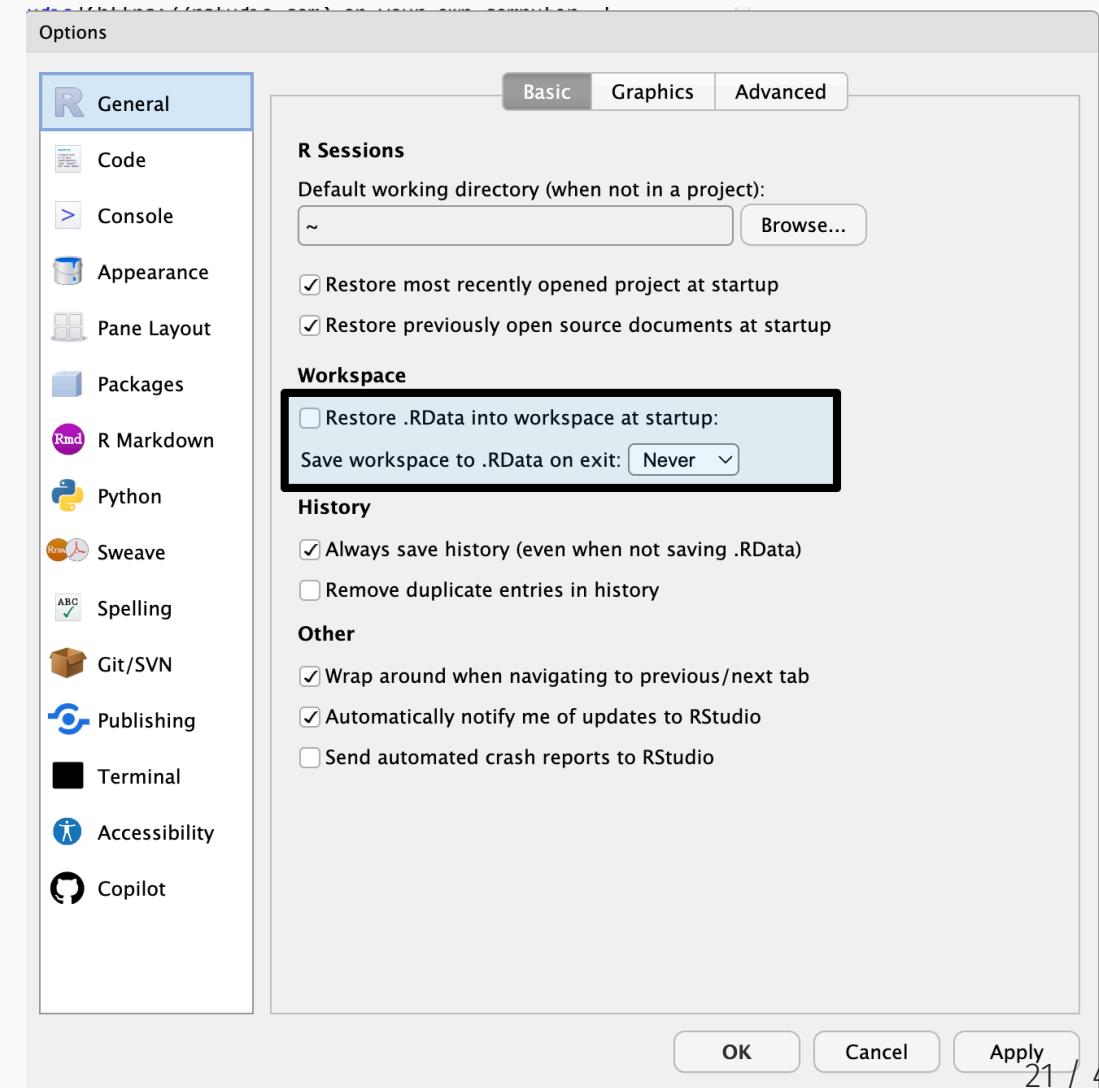
```
install.packages('tinytex')
tinytex::install_tinytex()
```

Clean Environment Between Sessions

By default RStudio will ask you to save your session when closing the application. Additionally, if the `.RData` file exists in the project directory, data saved in that file will be loaded into your session. *I highly recommend turning this off.* Doing so will ensure you have a clean environment everytime you start RStudio.

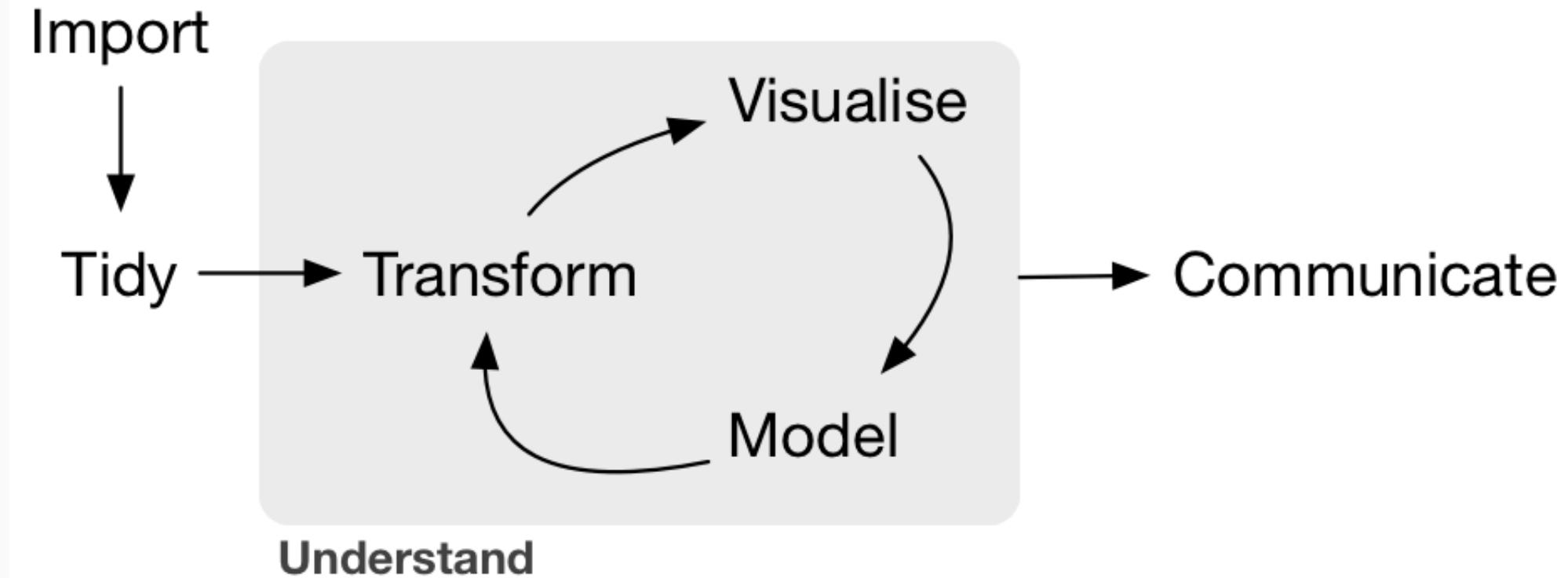
- In RStudio, click the `Tools` menu then `Global Options`
- Uncheck "Restore `.RData` into workspace at startup"
- For "Save workspace to `.RData` on exit" select "Never"

Why do this? Whenever you knit a document (e.g. Rmarkdow, Quarto) R will create a new, blank, session to knit the document. When working interactively it is easy to run commands out of order. It is always good to check your work by resetting your environment (click `Session --> Restart R`) and running your commands in order to ensure everything will work as intended.



Introduction to R

Workflow



Source: Wickham & Grolemund, 2017

Tidy Data

“**TIDY DATA** is a standard way of mapping the meaning of a dataset to its structure.”

—HADLEY WICKHAM

In tidy data:

- each variable forms a column
- each observation forms a row
- each cell is a single measurement

each column a variable

each row an observation

id	name	color
1	floof	gray
2	max	black
3	cat	orange
4	donut	gray
5	merlin	black
6	panda	calico

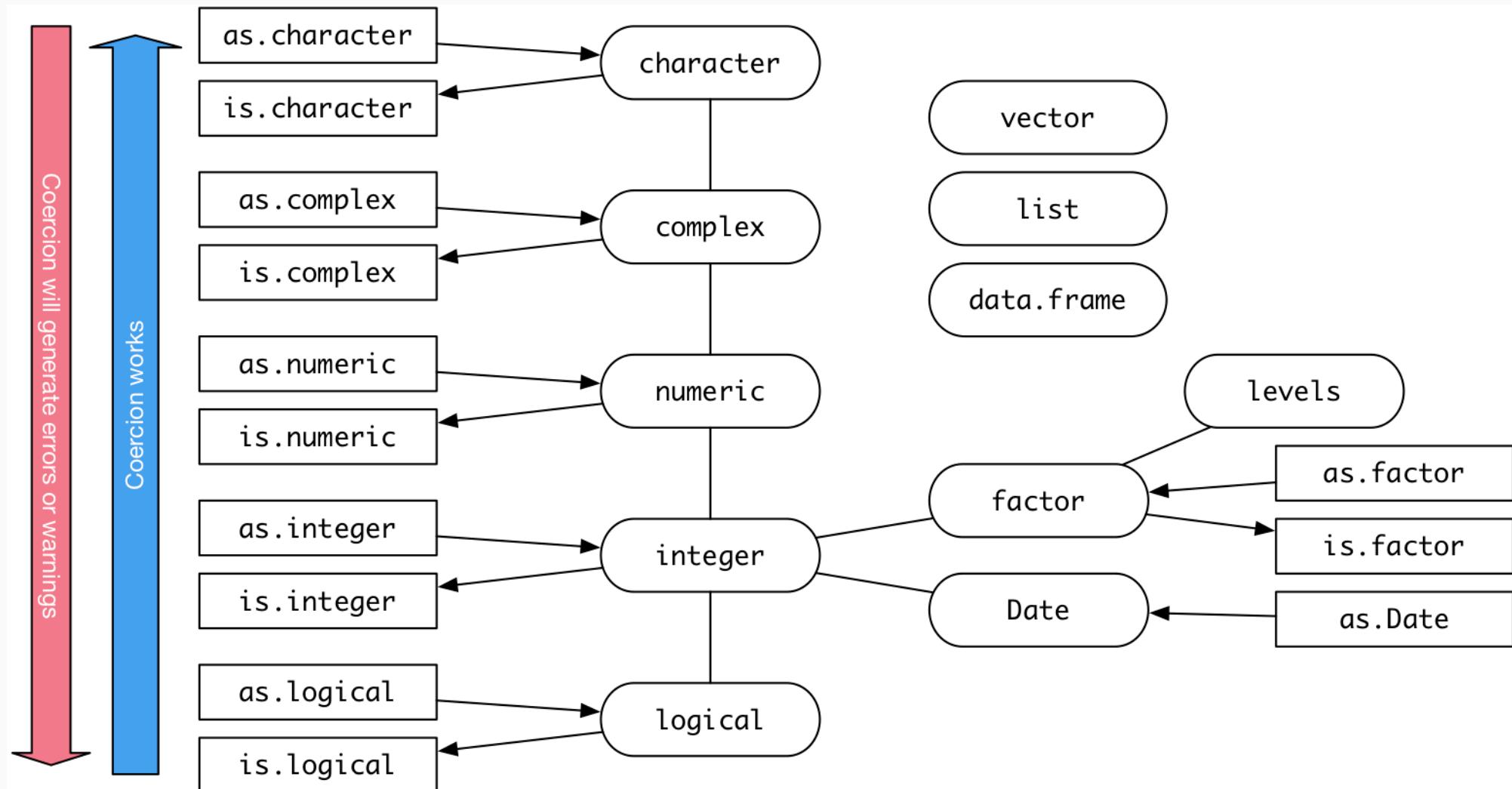
Wickham, H. (2014). Tidy Data. Journal of Statistical Software 59 (10). DOI: 10.18637/jss.v059.i10

Types of Data

- Numerical (quantitative)
 - Continuous
 - Discrete
- Categorical (qualitative)
 - Regular categorical
 - Ordinal



Data Types in R





About legosets

To install the `brickset` package:

```
remotes::install_github('jbryer/brickset')
```

To load the `legosets` dataset.

```
data('legosets', package = 'brickset')
```

The `legosets` data has 21546 observations of 36 variables.

```
names(legosets)
```

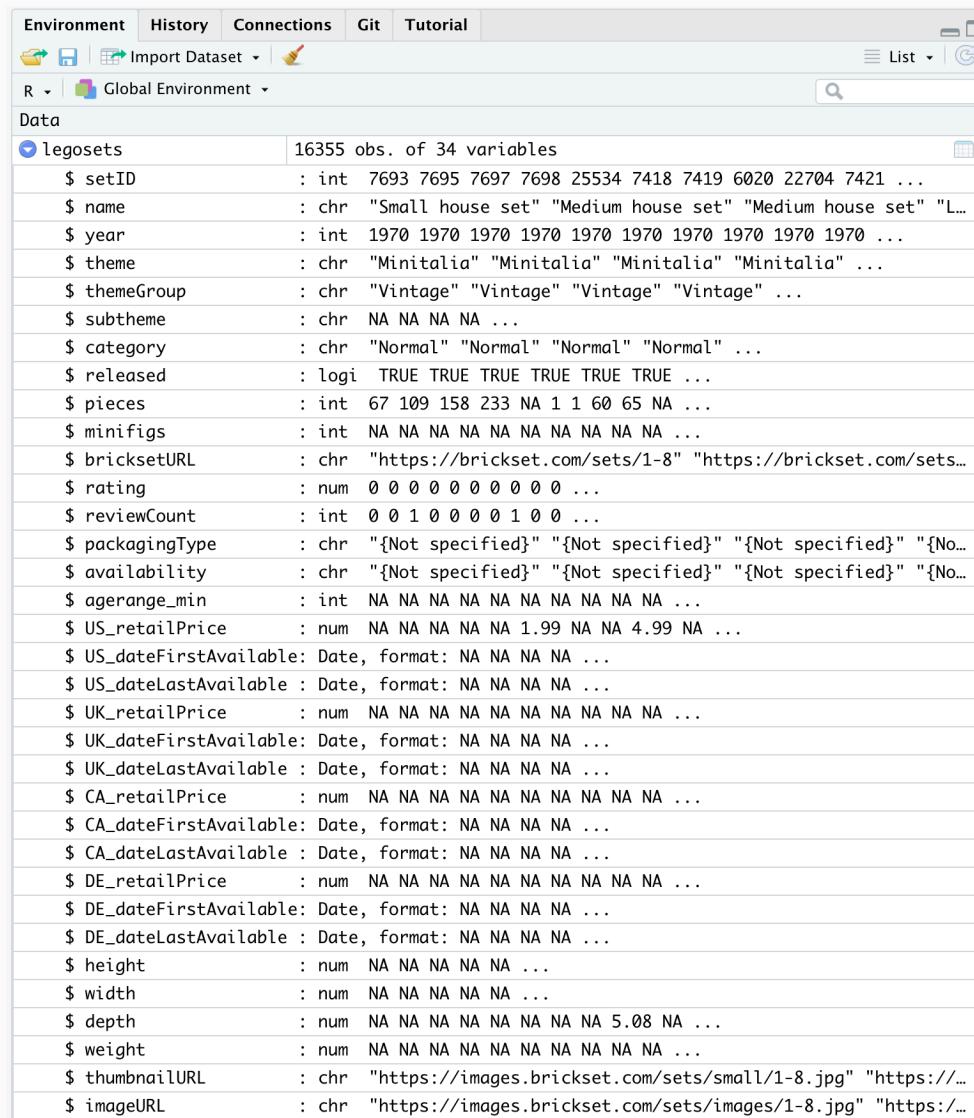
```
## [1] "setID"                 "number"                "numberVariant"  
## [4] "name"                  "year"                  "theme"  
## [7] "themeGroup"             "subtheme"              "category"  
## [10] "released"               "pieces"                "minifigs"  
## [13] "bricksetURL"            "rating"                "reviewCount"  
## [16] "packagingType"          "availability"          "agerange_min"  
## [19] "thumbnailURL"           "imageURL"              "US_retailPrice"  
## [22] "US_dateFirstAvailable" "US_dateLastAvailable" "UK_retailPrice"  
## [25] "UK_dateFirstAvailable" "UK_dateLastAvailable" "CA_retailPrice"  
## [28] "CA_dateFirstAvailable" "CA_dateLastAvailable" "DE_retailPrice"  
## [31] "DE_dateFirstAvailable" "DE_dateLastAvailable" "height"  
## [34] "width"                  "depth"                 "weight"
```

Structure (str)

str(legosets)

```
## 'data.frame': 21546 obs. of 36 variables:
## $ setID          : int 7693 7695 7697 7698 25534 ...
## $ number         : chr "1" "2" "3" "4" ...
## $ numberVariant : int 8 8 6 4 6 1 1 1 3 4 ...
## $ name           : chr "Small house set" "Medium house set" "Medium house set" "Large house set" ...
## $ year           : int 1970 1970 1970 1970 1970 1970 1970 1970 1970 ...
## $ theme          : chr "Minitalia" "Minitalia" "Minitalia" "Minitalia" ...
## $ themeGroup     : chr "Vintage" "Vintage" "Vintage" "Vintage" ...
## $ subtheme       : chr NA NA NA NA ...
## $ category       : chr "Normal" "Normal" "Normal" "Normal" ...
## $ released        : logi TRUE TRUE TRUE TRUE TRUE ...
## $ pieces          : int 67 109 158 233 NA 1 1 60 65 NA ...
## $ minifigs        : int NA NA NA NA NA NA NA NA NA ...
## $ bricksetURL    : chr "https://brickset.com/sets/1-8" "https://brickset.com/sets/2-8" "https://brickset.com/sets/3-6" "https://brickset.com/sets/4-4" ...
## $ rating          : num 0 0 0 0 0 0 0 0 0 ...
## $ reviewCount    : int 0 0 1 0 0 0 0 0 0 ...
## $ packagingType   : chr "{Not specified}" "{Not specified}" "{Not specified}" "{Not specified}" ...
## $ availability    : chr "{Not specified}" "{Not specified}" "{Not specified}" "{Not specified}" ...
## $ agerange_min    : int NA NA NA NA NA NA NA NA NA ...
## $ thumbnailURL   : chr "https://images.brickset.com/sets/small/1-8.jpg" "https://images.brickset.com/sets/small/2-8.jpg" "https://images.brickset.com/sets/small/3-6.jpg" "https://images.brickset.com/sets/small/4-4.jpg" ...
## $ imageURL        : chr "https://images.brickset.com/sets/images/1-8.jpg" "https://images.brickset.com/sets/images/2-8.jpg" "https://images.brickset.com/sets/images/3-6.jpg" "https://images.brickset.com/sets/images/4-4.jpg" ...
## $ US_retailPrice  : num NA NA NA NA NA NA NA NA NA ...
## $ US_dateFirstAvailable: Date, format: NA NA ...
## $ US_dateLastAvailable : Date, format: NA NA ...
## $ UK_retailPrice   : num NA NA NA NA NA NA NA NA NA ...
## $ UK_dateFirstAvailable: Date, format: NA NA ...
## $ UK_dateLastAvailable : Date, format: NA NA ...
## $ CA_retailPrice   : num NA NA NA NA NA NA NA NA NA ...
## $ CA_dateFirstAvailable: Date, format: NA NA ...
## $ CA_dateLastAvailable : Date, format: NA NA ...
## $ DE_retailPrice   : num NA NA NA NA NA NA NA NA NA ...
## $ DE_dateFirstAvailable: Date, format: NA NA ...
## $ DE_dateLastAvailable : Date, format: NA NA ...
## $ height           : num NA NA NA NA NA ...
## $ width            : num NA NA NA NA NA ...
```

RStudio Environment tab can help



The screenshot shows the RStudio interface with the 'Environment' tab selected. The main pane displays the 'legosets' dataset as a data frame with 16355 observations and 34 variables. The variables include setID, name, year, theme, themeGroup, subtheme, category, released, pieces, minifigs, bricksetURL, rating, reviewCount, packagingType, availability, agerange_min, US_retailPrice, US_dateFirstAvailable, US_dateLastAvailable, UK_retailPrice, UK_dateFirstAvailable, UK_dateLastAvailable, CA_retailPrice, CA_dateFirstAvailable, CA_dateLastAvailable, DE_retailPrice, DE_dateFirstAvailable, DE_dateLastAvailable, height, width, depth, weight, thumbnailURL, and imageURL. The data types for most variables are integers (int), characters (chr), or logicals (logi). Some variables like 'released' and 'reviewCount' have TRUE/FALSE values. The 'bricksetURL' and 'imageURL' variables contain URLs. The 'rating' variable is a numeric vector with many zeros. The 'reviewCount' variable has mostly zero values. The 'availability' variable has mostly NA values. The 'agerange_min' variable has mostly NA values. The 'US_retailPrice' and 'UK_retailPrice' variables have some non-NA values. The 'CA_retailPrice' and 'DE_retailPrice' variables have mostly NA values. The 'date' variables (dateFirstAvailable, dateLastAvailable) have NA values. The 'height', 'width', 'depth', and 'weight' variables have mostly NA values. The 'thumbnailURL' and 'imageURL' variables have URLs.

Table View

Show **10** entries Search:

	setID	name	year	theme	themeGroup	category	US_retailPrice	pieces	minifigs	rating
1	34275	Dump Truck	2023	Technic	Technical	Normal	12.99	177		3.8
2	524	Pen Darth Vader	2002	Gear	Miscellaneous	Gear				0
3	50401	Zane	2024	Ninjago	Action/Adventure	Other		11	1	0
4	26333	LEGO Star Wars: The Force Awakens - PlayStation 4	2016	Gear	Miscellaneous	Gear	69.99			0
5	6100	Gordon's Express	2007	Duplo	Pre-school	Normal	19.99	13	1	0
6	5051	Farm Animals Set	1986	Dacta	Educational	Normal		30		0
7	30791	Hogwarts Moment: Herbology Class	2021	Harry Potter	Licensed	Normal	29.99	233	3	3.8
8	3623	Galaxy Trekkor	1987	Space	Action/Adventure	Normal		29	1	3.2
9	49158	Brick Lunch Bag – Blue	2024	Gear	Miscellaneous	Gear	28			0
10	5397	Speedboat	2006	Creator	Model making	Normal		18		0

Showing 1 to 10 of 100 entries

Previous [1](#) [2](#) [3](#) [4](#) [5](#) ... [10](#) Next

Tidyverse vs Base R



R Syntax Comparison :: CHEAT SHEET

Dollar sign syntax

```
goal(data$x, data$y)
```

SUMMARY STATISTICS:

one continuous variable:
`mean(mtcars$mpg)`

one categorical variable:
`table(mtcars$cyl)`

two categorical variables:
`table(mtcars$cyl, mtcars$am)`

one continuous, one categorical:
`mean(mtcars$mpg[mtcars$cyl==4])`
`mean(mtcars$mpg[mtcars$cyl==6])`
`mean(mtcars$mpg[mtcars$cyl==8])`

PLOTTING:

one continuous variable:
`hist(mtcars$disp)`

`boxplot(mtcars$disp)`

one categorical variable:
`barplot(table(mtcars$cyl))`

two continuous variables:
`plot(mtcars$disp, mtcars$mpg)`

two categorical variables:
`mosaicplot(table(mtcars$am, mtcars$cyl))`

one continuous, one categorical:
`histogram(mtcars$disp[mtcars$cyl==4])`
`histogram(mtcars$disp[mtcars$cyl==6])`
`histogram(mtcars$disp[mtcars$cyl==8])`

`boxplot(mtcars$disp[mtcars$cyl==4])`
`boxplot(mtcars$disp[mtcars$cyl==6])`
`boxplot(mtcars$disp[mtcars$cyl==8])`

WRANGLING:

subsetting:
`mtcars[mtcars$mpg>30,]`

making a new variable:
`mtcars$efficient[mtcars$mpg>30] <- TRUE`
`mtcars$efficient[mtcars$mpg<30] <- FALSE`

Formula syntax

```
goal(y~x|z, data=data, group=w)
```

SUMMARY STATISTICS:

one continuous variable:
`mosaic::mean(~mpg, data=mtcars)`

one categorical variable:
`mosaic::tally(~cyl, data=mtcars)`

two categorical variables:
`mosaic::tally(cyl~am, data=mtcars)`

one continuous, one categorical:
`mosaic::mean(mpg~cyl, data=mtcars)`

tilde

PLOTTING:

one continuous variable:
`lattice::histogram(~disp, data=mtcars)`

`lattice::bwplot(~disp, data=mtcars)`

one categorical variable:
`mosaic::bargraph(~cyl, data=mtcars)`

two continuous variables:
`lattice::xyplot(mpg~disp, data=mtcars)`

two categorical variables:
`mosaic::bargraph(~am, data=mtcars, group=cyl)`

one continuous, one categorical:
`lattice::histogram(~disp|cyl, data=mtcars)`

`lattice::bwplot(cyl~disp, data=mtcars)`

The variety of R syntaxes give
you many ways to “say” the
same thing

read across the cheatsheet to see how different
syntaxes approach the same problem

Tidyverse syntax

```
data %>% goal(x)
```

SUMMARY STATISTICS:

one continuous variable:
`mtcars %>% dplyr::summarize(mean(mpg))`

one categorical variable:
`mtcars %>% dplyr::group_by(cyl) %>%
dplyr::summarize(n())`

the pipe

two categorical variables:
`mtcars %>% dplyr::group_by(cyl, am) %>%
dplyr::summarize(n())`

one continuous, one categorical:
`mtcars %>% dplyr::group_by(cyl) %>%
dplyr::summarize(mean(mpg))`

PLOTTING:
one continuous variable:
`ggplot2::qplot(x=mpg, data=mtcars, geom = "histogram")`

`ggplot2::qplot(y=disp, x=1, data=mtcars, geom="boxplot")`

one categorical variable:
`ggplot2::qplot(x=cyl, data=mtcars, geom="bar")`

two continuous variables:
`ggplot2::qplot(x=disp, y=mpg, data=mtcars, geom="point")`

two categorical variables:
`ggplot2::qplot(x=factor(cyl), data=mtcars, geom="bar") +
facet_grid(.~am)`

one continuous, one categorical:
`ggplot2::qplot(x=disp, data=mtcars, geom = "histogram") +
facet_grid(.~cyl)`

`ggplot2::qplot(y=disp, x=factor(cyl), data=mtcars,
geom="boxplot")`

WRANGLING:
subsetting:
`mtcars %>% dplyr::filter(mpg>30)`

making a new variable:
`mtcars <- mtcars %>%
dplyr::mutate(efficient = if_else(mpg>30, TRUE, FALSE))`



Data Wrangling Cheat Sheet

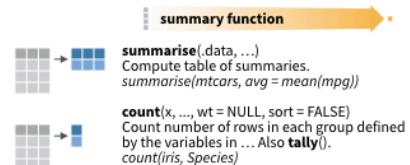
Data Transformation with dplyr :: CHEAT SHEET

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



Summarise Cases

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

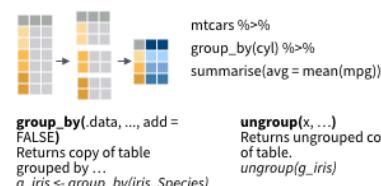


VARIATIONS

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

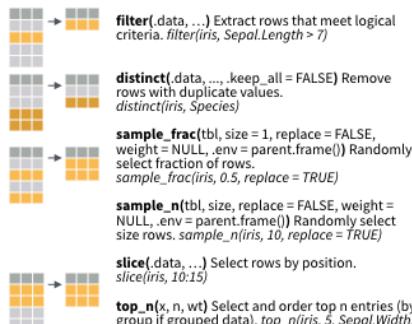
Use **group_by()** to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.



Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

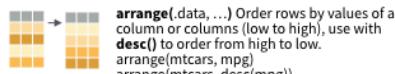


Logical and boolean operators to use with filter()

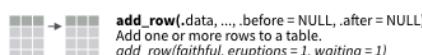
<	=<	is.na()	%in%		xor()
>	=>	is.na()	!	&	

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

ARRANGE CASES



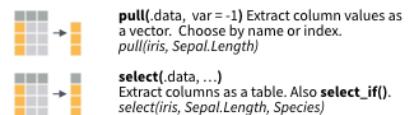
ADD CASES



Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

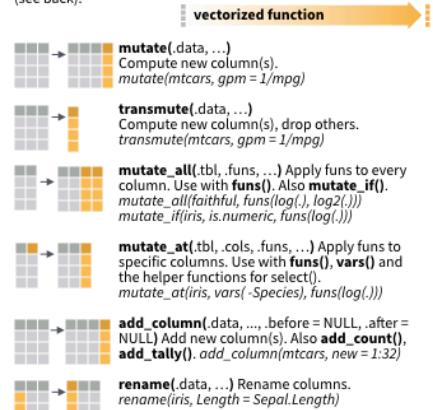


Use these helpers with **select()**, e.g. `select(iris, starts_with("Sepal"))`

contains (match)	num_range (prefix, range)	:	e.g. <code>mpg:cyl</code>
ends_with (match)	one_of (...)	,	e.g. <code>Species</code>
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).



Pipes %>% and |>



The pipe operator (`%>%`) introduced with the `magrittr` R package allows for the chaining of R operations. Base R has now added their own pipe operator (`|>`). They take the output from the left-hand side and passes it as the first parameter to the function on the right-hand side.



You can do this in two steps:

```
tab_out <- table(legosets$category)
prop.table(tab_out)
```

Or as nested function calls.

```
prop.table(table(legosets$category))
```

Using the pipe (`|>`) operator we can chain these calls in a what is arguably a more readable format:

```
legosets$category |> table() |> prop.table()
```

```
##          Book Collection Extended      Gear      Normal      Other
## 0.035087719 0.029889539 0.036248027 0.158869396 0.668894458 0.067205050
```

dplyr::filter()

KEEP ROWS THAT
satisfy
your CONDITIONS

keep rows from... this data... ONLY IF... type MATCHES "otter" AND site MATCHES "bay"
filter(df, type == "otter" & site == "bay")



A cartoon illustration featuring three characters: an orange circle with a smiling face, a purple circle with a question mark, and a green circle with a neutral face. They are positioned around a small map of a coastal area with a blue sea, a green landmass, and a yellow 'BAY' label. The purple character is pointing at a table, while the green character has a red 'X' drawn on its body.

	type	food	site
1	otter	urchin	bay
2	Shark	seal	channel
3	otter	abalone	bay
4	otter	crab	wharf

@allisonhorst

Logical Operators

- `!a` - TRUE if a is FALSE
- `a == b` - TRUE if a and be are equal
- `a != b` - TRUE if a and b are not equal
- `a > b` - TRUE if a is larger than b, but not equal
- `a >= b` - TRUE if a is larger or equal to b
- `a < b` - TRUE if a is smaller than be, but not equal
- `a <= b` - TRUE if a is smaller or equal to b
- `a %in% b` - TRUE if a is in b where b is a vector
- `a | b` - TRUE if a *or* b are TRUE
- `a & b` - TRUE if a *and* b are TRUE
- `isTRUE(a)` - TRUE if a is TRUE

%in% operator

```
vowels <- c('a', 'e', 'i', 'o', 'u')
```

```
letters %in% vowels
```

```
## [1] TRUE FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
## [17] FALSE FALSE FALSE FALSE TRUE FALSE FALSE
```

Which is equivalent to doing a bunch of or statements.

```
letters == 'a' | letters == 'e' | letters == 'i' | letters == 'o' | letters == 'u'
```

```
## [1] TRUE FALSE FALSE FALSE TRUE FALSE FALSE FALSE TRUE FALSE FALSE FALSE FALSE FALSE FALSE TRUE FALSE  
## [17] FALSE FALSE FALSE FALSE TRUE FALSE FALSE
```

We can use the `which()` function to determine the position of TRUE's.

```
letters %in% vowels |> which()
```

```
## [1] 1 5 9 15 21
```

Filter



dplyr

```
mylego <- legosets |> filter(themeGroup == 'Educational' & year > 2015)
```

Base R

```
mylego <- legosets[legosets$themeGroups == 'Educational' & legosets$year > 2015,]
```

```
nrow(mylego)
```

```
## [1] 121
```

Select



dplyr

```
mylego <- mylego |> select(setID, pieces, theme, availability, US_retailPrice, minifigs)
```

Base R

```
mylego <- mylego[,c('setID', 'pieces', 'theme', 'availability', 'US_retailPrice', 'minifigs')]
```

```
head(mylego, n = 4)
```

	setID	pieces	theme	availability	US_retailPrice	minifigs
## 1	26803	109	Education	{Not specified}	NA	6
## 2	26277	188	Education	Educational	94.95	NA
## 3	27742	160	Education	{Not specified}	NA	NA
## 4	26805	1000	Education	{Not specified}	NA	NA

Relocate



dplyr::**relocate()**
move COLUMNS around!

Default: move to FRONT
or move to
.before or .after
A SPECIFIED COLUMN!



Relocate



dplyr

```
mylego |> relocate(where(is.numeric), .after = where(is.character)) |> head(n = 3)
```

```
##      theme availability setID pieces US_retailPrice minifigs
## 1 Education {Not specified} 26803     109          NA       6
## 2 Education     Educational 26277     188        94.95      NA
## 3 Education {Not specified} 27742     160          NA      NA
```

Base R

```
mylego2 <- mylego[,c('theme', 'availability', 'setID', 'pieces', 'US_retailPrice', 'minifigs')]
head(mylego2, n = 3)
```

```
##      theme availability setID pieces US_retailPrice minifigs
## 1 Education {Not specified} 26803     109          NA       6
## 2 Education     Educational 26277     188        94.95      NA
## 3 Education {Not specified} 27742     160          NA      NA
```

Rename



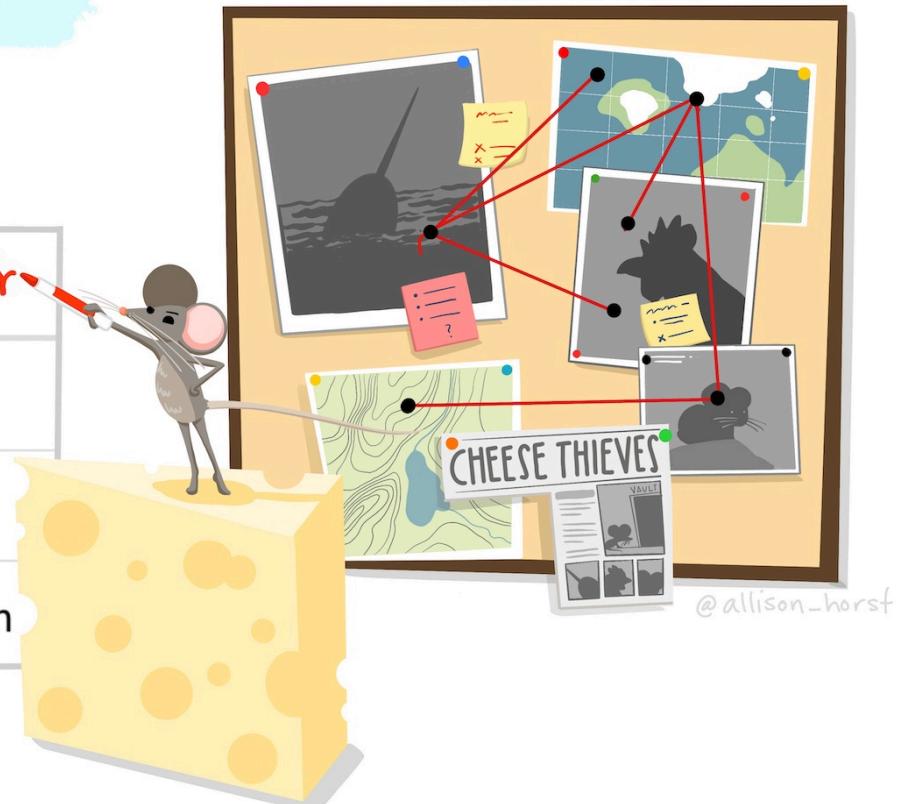
dplyr::rename()

RENAME COLUMNS*

df %>% rename(lair=site)

species nemesis	status	site lair
narwhal	unknown	ocean
chicken	active	coop
pika	active	mountain

*See `rename_with()` to rename using a function.



Rename



dplyr

```
mylego |> dplyr::rename(USD = US_retailPrice) |> head(n = 3)
```

```
## #> #> #> #>
```

	setID	pieces	theme	availability	USD	minifigs
## 1	26803	109	Education	{Not specified}	NA	6
## 2	26277	188	Education	Educational	94.95	NA
## 3	27742	160	Education	{Not specified}	NA	NA

Base R

```
names(mylego2)[5] <- 'USD'  
head(mylego2, n = 3)
```

```
## #> #> #> #>
```

	theme	availability	setID	pieces	USD	minifigs
## 1	Education	{Not specified}	26803	109	NA	6
## 2	Education	Educational	26277	188	94.95	NA
## 3	Education	{Not specified}	27742	160	NA	NA

Mutate



Mutate



dplyr

```
mylego |> filter(!is.na(pieces) & !is.na(US_retailPrice)) |>  
  mutate(Price_per_piece = US_retailPrice / pieces) |> head(n = 3)
```

```
## #> #> setID pieces theme availability US_retailPrice minifigs Price_per_piece  
## #> 1 26277 188 Education Educational 94.95 NA 0.5050532  
## #> 2 25949 280 Education Educational 224.95 NA 0.8033929  
## #> 3 25954 1 Education Educational 14.95 NA 14.9500000
```

Base R

```
mylego2 <- mylego[!is.na(mylego$US_retailPrice) & !is.na(mylego$Price_per_piece),]  
mylego2$Price_per_piece <- mylego2$Price_per_piece / mylego2$US_retailPrice  
head(mylego2, n = 3)
```

```
## [1] setID pieces theme availability US_retailPrice minifigs  
## [7] Price_per_piece  
## <0 rows> (or 0-length row.names)
```

Group By and Summarize

```
legosets |> group_by(themeGroup) |> summarize(mean_price = mean(US_retailPrice, na.rm = TRUE),  
                                sd_price = sd(US_retailPrice, na.rm = TRUE),  
                                median_price = median(US_retailPrice, na.rm = TRUE),  
                                n = n(),  
                                missing = sum(is.na(US_retailPrice)))
```

```
## # A tibble: 17 × 6  
##   themeGroup     mean_price    sd_price median_price      n missing  
##   <chr>          <dbl>       <dbl>      <dbl> <int>    <int>  
## 1 Action/Adventure    43.0        41.9      30.0  1620     845  
## 2 Art and crafts     41.0        53.0      20.0   104      9  
## 3 Basic              22.8        19.4      15.0   884    733  
## 4 Constraction       16.4        12.4      13.0   503    285  
## 5 Educational         185.        188.      138.   546    508  
## 6 Girls               35.8        24.0      23.0   240    227  
## 7 Historical          34.2        32.4      20.0   474    401  
## 8 Junior              22.0        10.1      20.0   228    165  
## 9 Licensed             55.5        72.9      35.0  3353   1245  
## 10 Miscellaneous       23.4        33.5      15.0  7151   4501  
## 11 Model making        86.2        99.4      50.0   919    422  
## 12 Modern day           39.8        36.8      30.0  2669   1594  
## 13 Pre-school            31.7        23.2      25.0  1613   1108  
## 14 Racing                26.4        26.8      15.0   266    176
```

Describe and Describe By

```
library(psych)
```

```
describe(legosets$US_retailPrice)
```

```
##    vars     n   mean      sd median trimmed    mad    min     max   range skew kurtosis     se
## X1     1 8674 42.26 59.75  24.99   30.25 22.24 1.49 999.99 998.5 4.97    40.38 0.64
```

```
describeBy(legosets$US_retailPrice, group = legosets$availability, mat = TRUE, skew = FALSE)
```

	item	group1	vars	n	mean	sd	median	min	max	range	se
## X11	1	{Not specified}	1	2004	27.76015	38.92310	19.99	1.49	789.99	788.5	0.8694780
## X12	2	Educational	1	12	217.03333	108.17617	232.45	14.95	399.95	385.0	31.2277699
## X13	3	Gift with Purchase at LEGO.com	1	0	NaN	NA	NA	Inf	-Inf	-Inf	NA
## X14	4	Insiders Reward	1	0	NaN	NA	NA	Inf	-Inf	-Inf	NA
## X15	5	LEGO exclusive	1	1268	65.12353	112.88247	14.99	1.99	999.99	998.0	3.1700555
## X16	6	LEGOLAND exclusive	1	2	4.99000	0.00000	4.99	4.99	4.99	0.0	0.0000000
## X17	7	Not sold	1	1	12.99000	NA	12.99	12.99	12.99	0.0	NA
## X18	8	Promotional	1	5	4.79000	0.83666	4.99	3.99	5.99	2.0	0.3741657
## X19	9	Promotional (Airline)	1	0	NaN	NA	NA	Inf	-Inf	-Inf	NA
## X110	10	Retail	1	5062	40.59401	40.96294	29.99	1.99	699.99	698.0	0.5757448
## X111	11	Retail - limited	1	319	63.40755	69.70365	39.99	2.49	449.99	447.5	3.9026552
## X112	12	Unknown	1	1	3.99000	NA	3.99	3.99	3.99	0.0	NA

Additional Resources

For data wrangling:

- `dplyr` website: <https://dplyr.tidyverse.org>
- R for Data Science book: <https://r4ds.had.co.nz/wrangle-intro.html>
- Wrangling penguins tutorial: <https://allisonhorst.shinyapps.io/dplyr-learnr/#section-welcome>
- Data transformation cheat sheet: <https://github.com/rstudio/cheatsheets/raw/master/data-transformation.pdf>

Good luck with the semester!



✉️ jason.bryer@yu.edu

🐱 @jbryer

📠 @jbryer@vis.social

slack dav5300spring2026.slack.com

🔗 spring2026.dav5300.net