

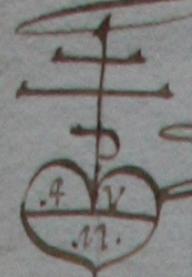
# vctrs: Creating custom vector classes with the vctrs package

Jesse Sadler  
Loyola Marymount University

@vivalosburros  
[jessesadler.com](http://jessesadler.com)  
[github.com/jessesadler](https://github.com/jessesadler)

Slides: [jessesadler.com/slides/RStudio2020.pdf](http://jessesadler.com/slides/RStudio2020.pdf)





Factuere van 6 baellenis witte lijwaeten № 97. 98. 99. 100. 101. 102. genyt  
als in maraine voor Ies de vogel vijt haerlen ouer Amsterdam op Sambores  
gesonden om van daer voorder ouer norenberg op Italien in sande van frank  
degeren beweccht te worden gonden als volgt

№ 97. goudende

Q.	1. metade duwt.	Q. 52 a &	35. becoopen.	£ 7. 11. 0
2	—	Q. 51 —	372 —	£ 7. 19. 93
3	—	Q. 512 —	40 —	£ 8. 11. 0
4	—	Q. 502 —	41 —	£ 8. 12. 62
5	—	Q. 51 —	44 —	£ 9. 7. 0
6	—	Q. 512 —	41 —	£ 8. 15. 112
7	—	Q. 51 —	41 —	£ 8. 14. 3
8	—	Q. 51 —	42 —	£ 8. 10. 6
9	—	Q. 502 —	43 —	£ 9. 0. 112
10	—	Q. 502 —	44 —	£ 9. 5. 2
11	—	Q. 51 —	45 —	£ 9. 11. 3
12	—	Q. 50 —	45 —	£ 9. 7. 6
13	—	Q. 50 —	46 —	£ 9. 11. 0
14	—	Q. 502 —	46 —	£ 9. 23. 7
15	—	Q. 51 —	46 —	£ 9. 15. 6
16	—	Q. 502 —	462 —	£ 9. 15. 0
17	—	Q. 50 —	47 —	£ 9. 15. 0
18	—	Q. 502 —	47 —	£ 9. 17. 92

80 104 lijwatten. — @ 914. becoopen versch. £ 164. 5. 102.

Per fmioglio ende oncosten

## Compound unit arithmetic

	£	s.	d.
Answer	<b>£134</b>	<b>15s.</b>	<b>11d.</b>
Unit total	132	53	35
Divide by base	-	53 / 20	35 / 12
Carried forward	2	2	-
Remainder	-	13	11

## Problem space

- Three separate units make up one value
- The units have non-decimal bases
- Need to use compound-unit arithmetic to normalize values
- The non-decimal bases differed by currency

## Compound unit arithmetic

	£	s.	d.
	28	15	8
	32	8	11
	54	18	7
	18	12	9
<b>Answer</b>	<b>£134</b>	<b>15s.</b>	<b>11d.</b>

	132	53	35
Unit total			
Divide by base	-	53 / 20	35 / 12
Carried forward	2	2	-
Remainder	-	13	11

## Simple normalization function

Fixed bases of 20s. and 12d.

```
# Normalize a numeric vector of length 3
normalize <- function(x) {
  pounds <- x[[1]] + ((x[[2]] + x[[3]]) %/%
    12) %/% 20)
  shillings <- (x[[2]] + x[[3]]) %/%
    12) %% 20
  pence <- x[[3]] %% 12

  c(pounds, shillings, pence)
}

normalize(c(132, 53, 35))
#> [1] 134 15 11
```

## Compound unit arithmetic

	£	s.	d.
	28	15	8
	32	8	11
	54	18	7
	18	12	9
<b>Answer</b>	<b>£134</b>	<b>15s.</b>	<b>11d.</b>
Unit total	132	53	35
Divide by base	-	53 / 20	35 / 12
Carried forward	2	2	-
Remainder	-	13	11

## Create an S3 class for non-decimal currencies

```
lsd <- function(x, bases = c(20, 12)) {  
  structure(x,  
            class = "lsd",  
            bases = bases)  
}  
  
lsd(c(134, 15, 11))  
#> [1] 134 15 11  
#> attr(,"class")  
#> [1] "lsd"  
#> attr(,"bases")  
#> [1] 20 12
```

## Compound unit arithmetic

	£	s.	d.
	28	15	8
	32	8	11
	54	18	7
	18	12	9

Answer

**£134 15s. 11d.**

Unit total

132            53            35

Divide by base

-             $53 / 20$              $35 / 12$

Carried forward

2            2            -

Remainder

-            13            11

## Create an S3 class for non-decimal currencies

```
lsd <- function(x, bases = c(20, 12)) {  
  structure(x,  
            class = "lsd",  
            bases = bases)  
}
```

```
lsd(c(134, 15, 11))  
#> [1] 134 15 11  
#> attr(,"class")  
#> [1] "lsd"  
#> attr(,"bases")  
#> [1] 20 12
```

# Create an S3 class for non-decimal currencies

## To-do list

Use lists instead of vectors to have multiple values

Change normalization method

What other methods do we need?

Print

Arithmetic operators

Concatenate

Subset Mathematical functions

Casting to other classes

Plots



# Create an S3 class for non-decimal currencies

## To-do list

Use lists instead of vectors to have multiple values

Change normalization method

What other methods do we need?

Print

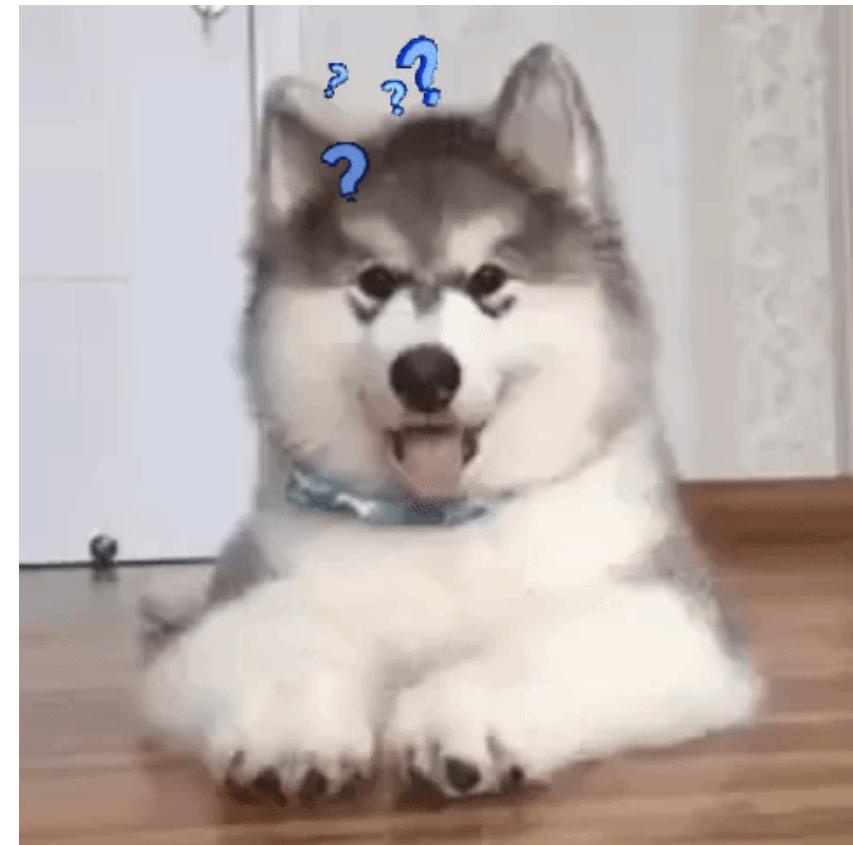
Concatenation

Subsetting

Mathematical functions

Casting to other classes

Plots





<https://vctrs.r-lib.org>



# Goals of vctrs

- Type stability
- Size stability
- Make it easier to build new S3 classes



# What do you get by using vctrs?

- Clear development path for creating an S3 class
- Consistency with base R functionality
- Integration with the tidyverse

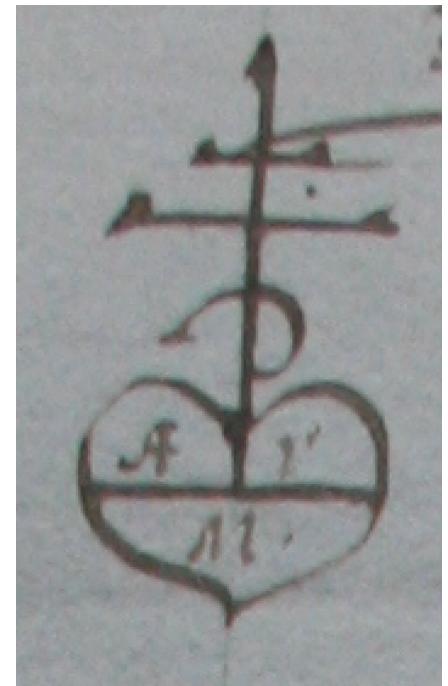
# Goals for the talk

- Why you might want to create your own S3 class
- Why you should use vctrs
- Point you to how you can do it

# debvctrs

Why and how to use vctrs

- debvctrs example package on GitHub:
  - [github.com/jessesadler/debvctrs](https://github.com/jessesadler/debvctrs)
- Simplified version of debkeepr:
  - [jessesadler.github.io/debkeepr](https://jessesadler.github.io/debkeepr)
- Step-by-step guide to building S3-vector classes with vctrs
  - Use in tandem with vctrs S3 vignette
  - <https://vctrs.r-lib.org/articles/s3-vector>



# Creating S3 classes with vctrs

1. Creation of the class
2. Coercion: implicit transformation of a class: `c()`
3. Casting: explicit transformation of a class: `as.numeric()`
4. Equality and comparison: `>`, `<`, `==`, etc.
5. Mathematical functions: `sum()`, `mean()`, etc.
6. Arithmetic operations: `+`, `-`, `*`, `/`, etc.

# Creating S3 classes with vctrs based on double vector

1. Creation of the class
2. Coercion: implicit transformation of a class: `c()`
3. Casting: explicit transformation of a class: `as.numeric()`
4. Equality and comparison: `>`, `<`, `==`, etc.
5. Mathematical functions: `sum()`, `mean()`, etc.
6. Arithmetic operations: `+`, `-`, `*`, `/`, etc.

A screenshot of a file explorer window. The path is Home > Documents > R > debvctrs > R. The table lists files in the debvctrs directory:

	Name	Size
	..	
	01.1-decimal-class.R	4.2 KB
	01.2-lsd-class.R	4.3 KB
	01.3-checks.R	2.3 KB
	02-coercion.R	3.4 KB
	03-casting.R	8.9 KB
	04-comparison-lsd.R	1.1 KB
	05-mathematical-funcs.R	3.3 KB
	06-arithmetic-ops.R	7.2 KB
	debvctrs-package.R	918 B
	helper-convert-attr.R	2.4 KB
	helper-normalize.R	3.7 KB
	utils.R	476 B

# debvctrs R scripts

[github.com/jessesadler/debvctrs](https://github.com/jessesadler/debvctrs)

## Compound unit arithmetic

	£	s.	d.
Answer	<b>£134</b>	<b>15s.</b>	<b>11d.</b>
Unit total	132	53	35
Divide by base	-	53 / 20	35 / 12
Carried forward	2	2	-
Remainder	-	13	11

## Problem space

- Three separate units make up one value
- The units have non-decimal bases
- Need to use compound-unit arithmetic to normalize values
- The non-decimal bases differed by currency

# Design principles

## deb\_lsd

- A class that maintains the tripartite structure of non-decimal currencies
- Tracks the bases of shillings and pence units
- Vectors with different bases cannot be combined

## deb\_decimal

- Decimalized class as fall back
- Tracks the bases of shillings and pence units
- Vectors with different bases cannot be combined
- Choose and track unit represented by decimalized class
- Vectors with different units can be combined but need coercion path

# 1. Creation

01.1-decimal-class.R, 01.2-lsd-class.r, and 01.3-check.R

1. Constructor: `new_lsd()` and `new_decimal()`
2. Helper: `deb_lsd()` and `deb_decimal()`
3. Formally declare S3 class: `setOldClass()`
4. Attribute access: `deb_bases()` and `deb_unit()`
5. Class check: `deb_is_lsd()` and `deb_is_decimal()`
6. Format method
7. Abbreviated name type

# 1. Creation

01.1-decimal-class.R, 01.2-lsd-class.r, and 01.3-check.R

## deb\_lsd()

```
# 1. Constructor
new_lsd <- function(l = double(),
                      s = double(),
                      d = double(),
                      bases = c(20L, 12L)) {

vctrs::new_rcrd(list(l = l, s = s, d = d),
                 bases = bases,
                 class = "deb_lsd")
}
```

## deb\_decimal()

```
# 1. Constructor
new_decimal <- function(x = double(),
                         unit = c("l", "s", "d"),
                         bases = c(20L, 12L)) {

vctrs::new_vctr(.data = x,
                unit = unit,
                bases = bases,
                class = "deb_decimal",
                inherit_base_type = TRUE)
}
```

# 1. Creation

01.1-decimal-class.R, 01.2-lsd-class.r, and 01.3-check.R

`deb_lsd()`

Arguments

`deb_decimal()`

```
# 1. Constructor
new_lsd <- function(l = double(),
                     s = double(),
                     d = double(),
                     bases = c(20L, 12L)) {

vctrs::new_rcrd(list(l = l, s = s, d = d),
                bases = bases,
                class = "deb_lsd")
}
```

```
# 1. Constructor
new_decimal <- function(x = double(),
                         unit = c("l", "s", "d"),
                         bases = c(20L, 12L)) {

vctrs::new_vctr(.data = x,
                unit = unit,
                bases = bases,
                class = "deb_decimal",
                inherit_base_type = TRUE)
}
```

Creation of class

# Structure of the classes

`deb_lsd()`

`deb_decimal()`

```
deb_lsd(l = c(17, 32, 18),  
        s = c(16, 7, 12),  
        d = c(6, 9, 3))
```

```
#> <deb_lsd[3]>  
#> [1] 17:16s:6d 32:7s:9d  
#> [3] 18:12s:3d  
#> # Bases: 20s 12d
```

```
deb_decimal(x = c(17.8250,  
                 32.3875,  
                 18.6125))
```

```
#> <deb_decimal[3]>  
#> [1] 17.8250 32.3875  
#> [3] 18.6125  
#> # Unit: pounds  
#> # Bases: 20s 12d
```

# Structure of the classes

`deb_lsd()`

record-style vector

```
deb_lsd(l = c(17, 32, 18),  
        s = c(16, 7, 12),  
        d = c(6, 9, 3))
```

```
#> <deb_lsd[3]>  
#> [1] 17:16s:6d 32:7s:9d  
#> [3] 18:12s:3d  
#> # Bases: 20s 12d
```

Bases attribute

`deb_decimal()`

double vector

```
deb_decimal(x = c(17.8250,  
                  32.3875,  
                  18.6125))
```

```
#> <deb_decimal[3]>  
#> [1] 17.8250 32.3875  
#> [3] 18.6125  
#> # Unit: pounds  
#> # Bases: 20s 12d
```

Unit attribute

Printing methods

## Both work natively in a tibble

```
tibble(lsd = deb_lsd(l = c(17, 32, 18),  
                      s = c(16, 7, 12),  
                      d = c(6, 9, 3)),  
      decimal = deb_decimal(x = c(17.8250,  
                                 32.3875,  
                                 18.6125)))  
  
#> # A tibble: 3 x 2  
#>       lsd     decimal  
#>   <l[20s:12d]> <l[20s:12d]>  
#> 1    17:16s:6d    17.8250  
#> 2    32:7s:9d    32.3875  
#> 3    18:12s:3d   18.6125
```

# Coercion and casting with vctrs

1. Creation of the class
2. Coercion: implicit transformation of a class: `c()`
3. Casting: explicit transformation of a class: `as.numeric()`
4. Equality and comparison: `>`, `<`, `==`, etc.
5. Mathematical functions: `sum()`, `mean()`, etc.
6. Arithmetic operations: `+`, `-`, `*`, `/`, etc.

# Coercion and casting workflow

## 1. Boilerplate

- Define method for class
- Default method for class for incompatible inputs

## 2. Methods within the class

## 3. Methods with compatible classes

# Coercion and casting

- Coercion looks for the common type:  
`vec_ptype2(x, y)`
- Casting does the actual transformation:  
`vec_cast(x, to)`
- Casting makes comparison between classes possible

## Design choices: coercion hierarchy

Define possibilities and implement  
hierarchy with `vec_ptype2(x, y)`

`double()` → `deb_decimal()` → `deb_lsd()`

# Implementation with casting

Example of deb\_decimal() to deb\_lsd()

```
vec_cast.deb_lsd.deb_decimal <- function(x, to, ...) {  
  bases_equal(x, to) # ensure that bases are equal  
  # if else depending on the unit  
  if (deb_unit(x) == "l") {  
    lsd <- deb_lsd(x, 0, 0, bases = deb_bases(x))  
  } else if (deb_unit(x) == "s") {  
    lsd <- deb_lsd(0, x, 0, bases = deb_bases(x))  
  } else if (deb_unit(x) == "d") {  
    lsd <- deb_lsd(0, 0, x, bases = deb_bases(x))  
  }  
  # Normalize the deb_lsd() vector  
  deb_normalize(lsd)  
}
```

# Put it all together

```
# Combine multiple types
c(deb_lsd(134, 15, 11), deb_decimal(14.875), 28.525)
#> <deb_lsd[3]>
#> [1] 134:15s:11d 14:17s:6d    28:10s:6d
#> # Bases: 20s 12d

# Compare different types
deb_decimal(3255, unit = "d") > deb_lsd(15, 13, 4)
#> [1] FALSE

# Arithmetic with different types
deb_decimal(3255, unit = "d") + deb_lsd(15, 13, 4)
#> <deb_lsd[1]>
#> [1] 29:4s:7d
#> # Bases: 20s 12d
```



# You can create your own S3 vector



- Extend the capabilities of R to fit your own needs
- vctrs provides a clear development path

Jesse Sadler

Twitter: @vivalosburros

website: jessesadler.com

GitHub: [github.com/jessesadler](https://github.com/jessesadler)

## Resources

- Slides: [jessesadler.com/slides/RStudio2020.pdf](https://jessesadler.com/slides/RStudio2020.pdf)
- debvctrs: [github.com/jessesadler/debvctrs](https://github.com/jessesadler/debvctrs)
- debkeepr: [jessesadler.github.io/debkeepr](https://jessesadler.github.io/debkeepr)
- vctrs website: [vctrs.r-lib.org](https://vctrs.r-lib.org)
  - The S3 vignette is particularly helpful
- Hadley Wickham, *Advanced R*: Chapter 13: S3