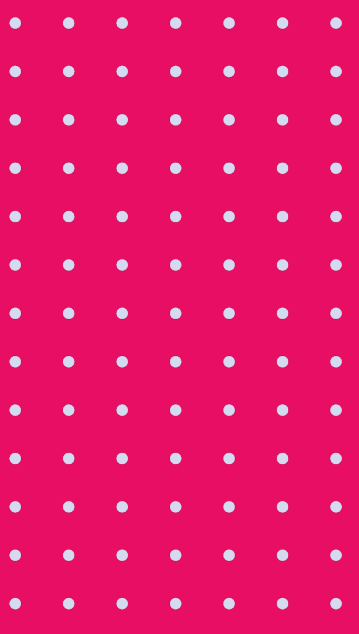


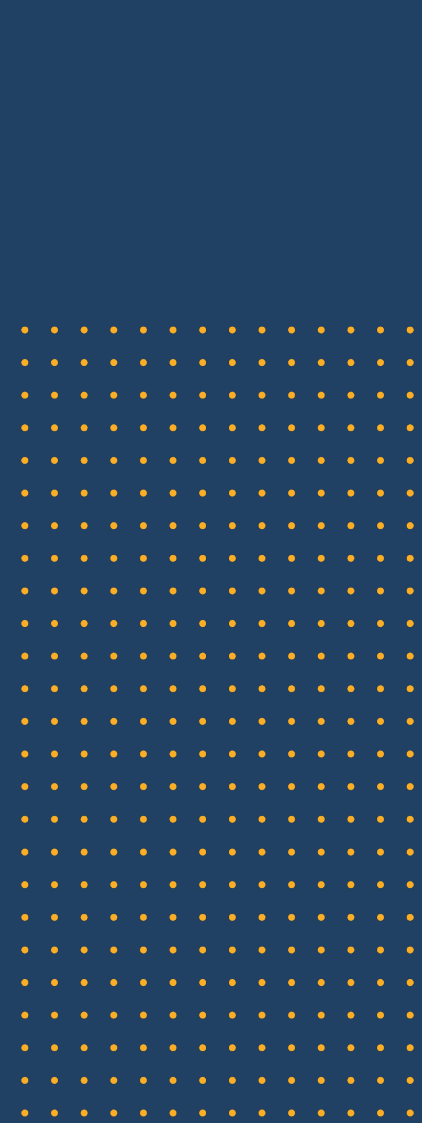


Functional Programming: What, why and how?

Tom Ratford – Veramed

PHUSE EU Connect 2022





Theory

THEORY

Imperative Programming

- Imperative programming focuses on the concept of state
- A change of state caused by code is known as a *side effect*

SAS

```
data mydata;  
  set olddata;  
  a=1;  
run;
```

SAS

```
proc sort data=olddata;  
  by ordvar;  
run;
```

THEORY

Functional Programming

- Functional programming rejects the idea of state and instead focuses on input and output
- A language is considered functional if it directs or forces you to create a solution that does not rely on state

maths

We want to apply a function g , followed by another function f , on our data denoted x .

This is a composition of f and g

$$f \circ g(x) = f(g(x))$$

THEORY

First class functions

A first class function essentially means that a function can be considered as a value like 1 or "Hello"

```
R  
nchar
```

```
## function (x, type = "chars", allowNA = FALSE, keepNA = NA)  
## .Internal(nchar(x, type, allowNA, keepNA))  
## <bytecode: 0x0000000024bd72a8>  
## <environment: namespace:base>
```

THEORY

First class functions

A first class function essentially means that a function can be considered as a value like 1 or "Hello"

R

```
strlength <- nchar  
strlength
```

```
## function (x, type = "chars", allowNA = FALSE, keepNA = NA)  
## .Internal(nchar(x, type, allowNA, keepNA))  
## <bytecode: 0x0000000024bd72a8>  
## <environment: namespace:base>
```

THEORY

First class functions

A first class function essentially means that a function can be considered as a value like 1 or "Hello"

R

```
nchar("hello")
```

```
## [1] 5
```

R

```
strlength("hello")
```

```
## [1] 5
```

THEORY

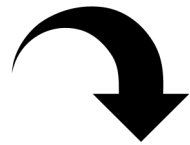
First class functions

A first class function essentially means that a function can be considered as a value like 1 or "Hello"

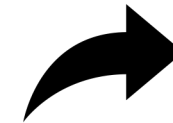
```
R  
myFunc <- function(...) {  
  ...  
}
```

THEORY

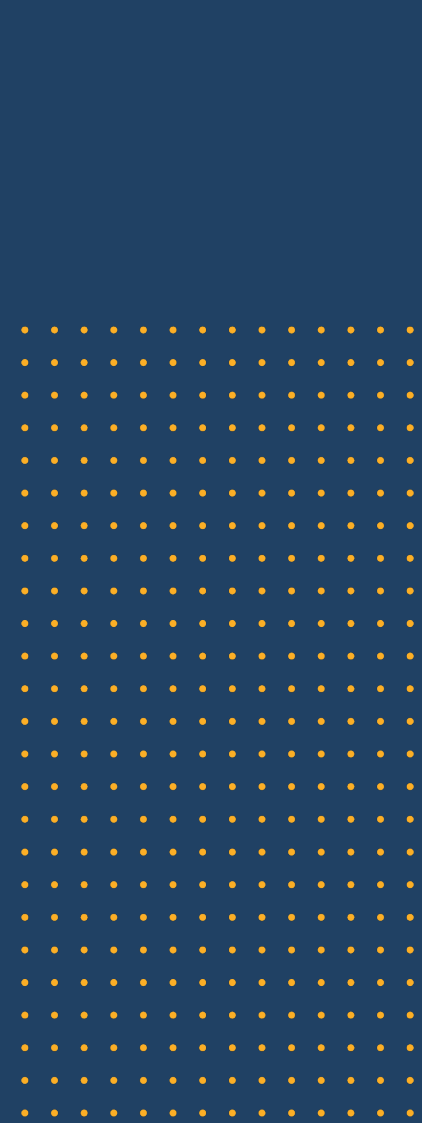
Higher order function



Takes a function
as an argument



Returns a function
as its result



Applications

APPLICATIONS: RETURNS A FUNCTION

Rounding in an ADLB table

USUBJID	PARAMCD	PARCAT1	AVAL	AVALC	AVISIT
101001-101101	LB00185	Cat 2	3.008100	3.0081	Day 1
101001-101101	LB00185	Cat 2	2.952000	2.952	Day 2
101001-101101	LB00185	Cat 2	3.799200	3.7992	Day 3
101001-101101	LB00185	Cat 2	3.398850	3.39885	Day 4
101001-101101	LB00185	Cat 2	1.062292	1.062292	Day 5
101001-101101	LB00192	Cat 1	8.304605	8.304605	Day 1

Rounding in an ADLB table

R

```
round_by_paramcd <- function(group) {  
  str_ext <- nchar(str_extract(group, "\\..*")) - 1  
  str_ext[is.na(str_ext)] <- 0  
  dps <- round(mean(str_ext), 0)  
  
  function(x) {  
    round(x, digits = dps) %>%  
      format(., nsmall = dps)  
  }  
}
```

Rounding in an ADLB table

R

```
round_by_paramcd <- function(group) {  
  str_ext <- nchar(str_extract(group, "\\..*")) - 1  
  str_ext[is.na(str_ext)] <- 0  
  dps <- round(mean(str_ext), 0)  
  
  function(x) {  
    round(x, digits = dps) %>%  
      format(., nsmall = dps)  
  }  
}
```

Rounding in an ADLB table

R

```
round_by_paramcd <- function(group) {  
  str_ext <- nchar(str_extract(group, "\\..*")) - 1  
  str_ext[is.na(str_ext)] <- 0  
  dps <- round(mean(str_ext), 0)  
  
  function(x) {  
    round(x, digits = dps) %>%  
      format(., nsmall = dps)  
  }  
}
```

Rounding in an ADLB table

R

```
round_by_paramcd <- function(group) {  
  str_ext <- nchar(str_extract(group, "\\..*")) - 1  
  str_ext[is.na(str_ext)] <- 0  
  dps <- round(mean(str_ext), 0)  
  
  function(x) {  
    round(x, digits = dps) %>%  
      format(., nsmall = dps)  
  }  
}
```

APPLICATIONS: RETURNS A FUNCTION

Rounding in an ADLB table

```
R
adlb %>%
  group_by(PARAMCD) %>%
  summarise(
    mean = round_by_paramcd(AVALC) (mean(AVAL))
  )
```

PARAMCD	mean
LB00185	1.82
LB00192	8.02757
LB00283	3.9978



Applying attributes to columns in a data frame

```
R
x <- 1:5
walk(x, ~ . + 1) %>%
  print
```

```
## [1] 1 2 3 4 5
```

```
R
map_dbl(x, ~ . + 1) %>%
  print
```

```
## [1] 2 3 4 5 6
```



Applying attributes to columns in a data frame

R

```
walk(x, function(x) {  
  attr(x, "PHUSE") <- "Hi EU Connect!"  
})  
attributes(x)
```

```
## $PHUSE  
## [1] "Hi EU Connect!"
```



Applying attributes to columns in a data frame

R

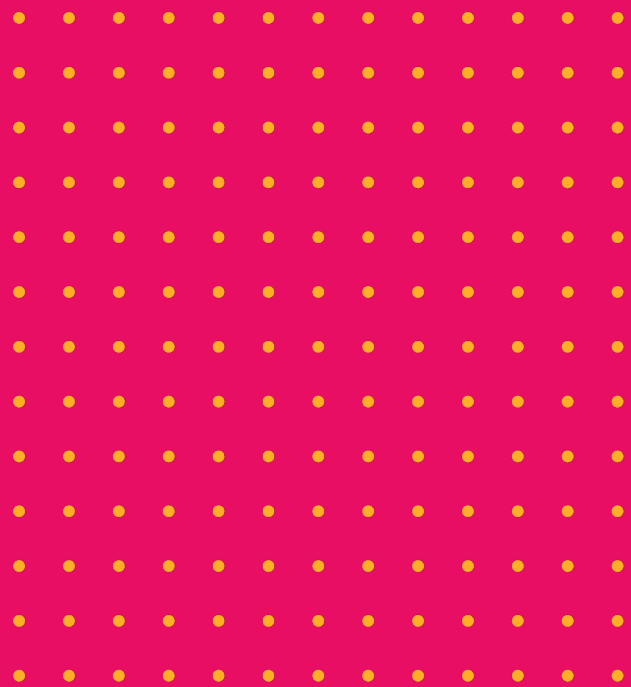
```
labels <- c(USUBJID="Unique Subject Identifier",
            PARAMCD="Parameter Code",
            PARCAT="Parameter Category",
            AVAL="Value",
            AVALC="Value (Character)",
            AVISIT="Analysis Visit")

walk2(
  names(labels),
  labels,
  function(x,y) {
    attr(ADLB[[x]], "label") <- y
  }
)
```

APPLICATIONS: FUNCTION AS AN ARGUMENT

Rounding in an ADLB table

USUBJID Unique Subject Identifier	PARAMCD Parameter Code	PARCAT1 Parameter Category	AVAL Value	AVALC Value (Character)	AVISIT Analysis Visit
101001-101101	LB00185	Cat 2	3.008100	3.0081	Day 1
101001-101101	LB00185	Cat 2	2.952000	2.952	Day 2
101001-101101	LB00185	Cat 2	3.799200	3.7992	Day 3
101001-101101	LB00185	Cat 2	3.398850	3.39885	Day 4
101001-101101	LB00185	Cat 2	1.062292	1.062292	Day 5
101001-101101	LB00192	Cat 1	8.304605	8.304605	Day 1



Contact me:

tom.ratford@veramed.co.uk

Thank you!

Find us at **BOOTH 26**

Demos:

- Data Visualisation
 - Automation
 - Cloud SCE