R Functions

# Dealing with failure

# The map functions fail hard

```
> log_list <- map(long_list, log)
Error in .f(.x[[i]], ...) : non-numeric argument to mathematical
function
```

# Introducing `safely()`

```
> log_list <- map(long_list, log)
Error in .f(.x[[i]], ...) : non-numeric argument to mathematical
function
```

```
> log_list <- map(long_list, safely(log))
> str(log_list)
List of 11
 $ :List of 2
  ..$ result: num [1:10] 1.386 1.386 2.079 1.946 0.693 ...
  ..$ error : NULL
 $ :List of 2
  ..$ result: NULL
  ..$ error : "Non-numeric input"
...
```

# `safely()` takes a function and returns a function

```
> library(purrr)

> safe_log <- safely(log)

> safe_log
function (...)
capture_error(.f(...), otherwise)
<environment: 0x101a44948>
```

# safely() never fails

```
> safe_log(10)
$result
[1] 2.302585

$error
NULL

> safe_log("a")
$result
NULL

$error
<simpleError in .f(...): non-numeric argument to mathematical
function>
```

**A list with two elements: result and error**

# Other adverbs for unusual output

- `safely()` captures the successful result or the error, always returns a list
- `possibly()` always succeeds, you give it a default value to return when there is an error
- `quietly()` captures printed output, messages, and warnings instead of capturing errors

R Functions

# Let’s practice!

R Functions

# Maps over multiple arguments

# Drawing samples from a Normal

```
> rnorm(5)
> rnorm(10)
> rnorm(20)

> map(list(5, 10, 20), rnorm)
```

```
rnorm(n, mean = 0, sd = 1)
```

# Use `map2()` to iterate over two arguments

```
> rnorm(5, mean = 1)
> rnorm(10, mean = 5)
> rnorm(20, mean = 10)

> map2(list(5, 10, 20), list(1, 5, 10), rnorm)
```

```
map2(.x, .y, .f, ...)
```

# pmap() to iterate over many arguments

```
> rnorm(5, mean = 1, sd = 0.1)
> rnorm(10, mean = 5, sd = 0.5)
> rnorm(20, mean = 10, sd = 0.1)

> pmap(list(n =    list(5, 10, 20),
            mean = list(1, 5, 10),
            sd =   list(0.1, 0.5, 0.1)), rnorm)
```

```
pmap(.l, .f, ...)
```

# invoke_map() to iterate over functions

```
> rnorm(5)
> runif(5)
> rexp(5)

> invoke_map(list(rnorm, runif, rexp), n = 5)
```

```
invoke_map(.f, .x = list(NULL), ...)
```

# Mapping over many arguments

- `map2()` - iterate over two arguments
- `pmap()` - iterate over many arguments
- `invoke_map()` - iterate over functions and arguments
- Like `map()`, each has a whole family of functions: `map2_dbl()`, `map2_lgl()`, `pmap_dbl()`, etc.

R Functions

# Let’s practice!

R Functions

# Maps with side effects

# Side effects

- Describe things that happen beyond the results of a function
- Examples include: printing output, plotting, and saving files to disk
- `walk()` works just like `map()`, but is designed for functions called for their side effects

# Introducing walk()

```
> x <- list(1, "a", 3)

> x %>% walk(print)
[1] 1
[1] "a"
[1] 3

> library(ggplot2)
> plots <- cyl %>%
    map(~ ggplot(., aes(mpg, wt)) + geom_point())

> paths <- paste0(names(plots), ".pdf")

> walk2(paths, plots, ggsave)
```

# Return value of `walk()`

```
> x <- list(1, "a", 3)

> out <- x %>% walk(print)
[1] 1
[1] "a"
[1] 3

> str(out)
List of 3
 $ : num 1
 $ : chr "a"
 $ : num 3
```

# walk() in a pipeline

```
> lengths <- x %>% walk(print) %>% map_dbl(length)
[1] 1
[1] "a"
[1] 3

> lengths
[1] 1 1 1
```

R Functions

# Let’s practice!