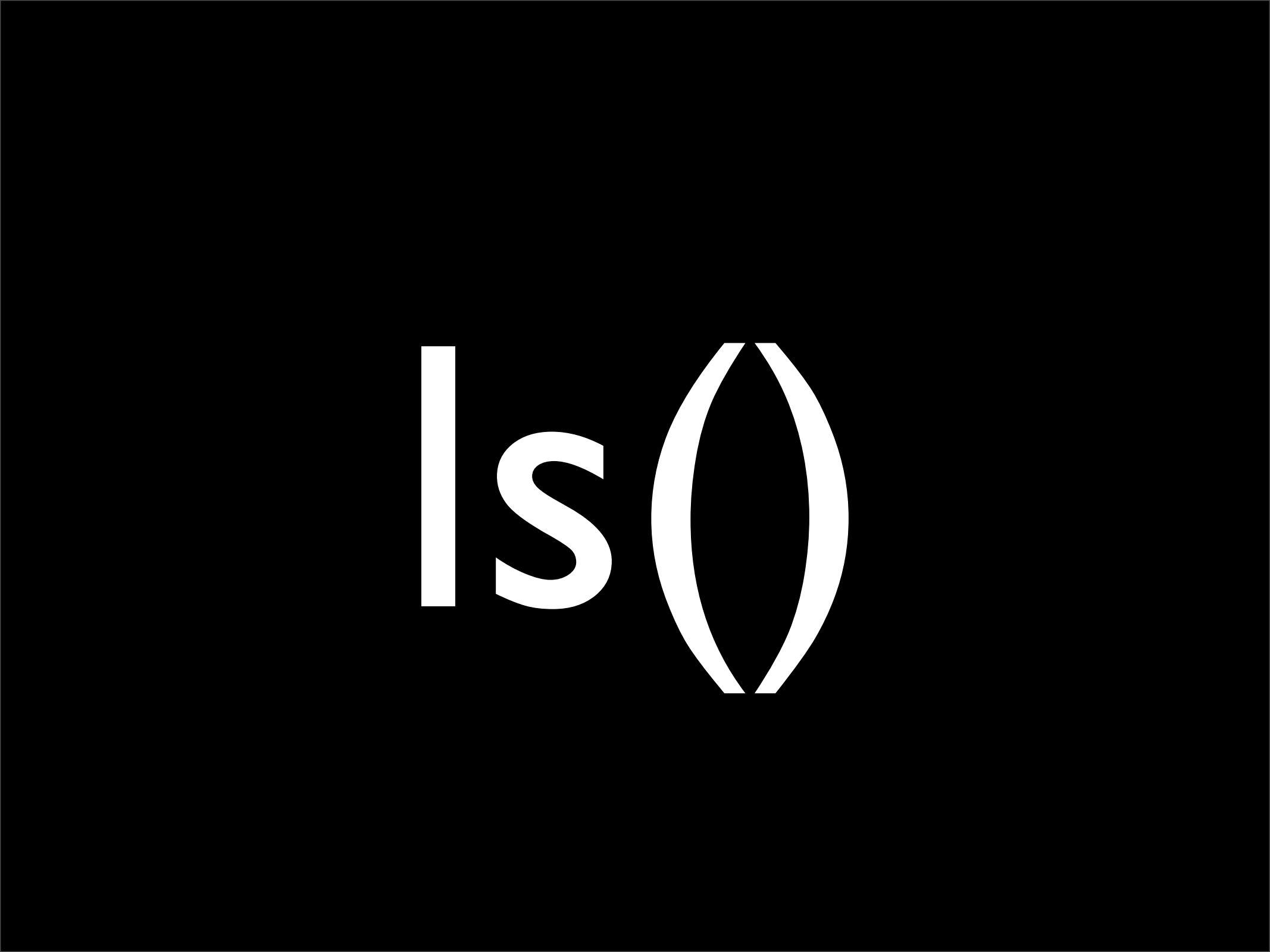
Being lazy with R

An introduction to functional programming

Functional programming

- Data-centric
- Less code (so hopefully fewer bugs and quicker to write)
- NO side effects

```
load(url("http://had.co.nz/de.zip"))
```


Is()


str(de)


deseasonalisation

```
str(de$ozone)
```

```
str(de$ozone)
de$ozone[1,1,]
```

```
str(de$ozone)
de$ozone[1,1,]
de$ozone[, , 1]
```

```
str(de$ozone)
de$ozone[1,1,]
de$ozone[, , 1]
de$ozone[1,1,1]
```

```
str(de$ozone)
de$ozone[1,1,]
de$ozone[, , 1]
de$ozone[1,1,1]
```

```
o <- de$ozone[1,1,]
plot(o, type="l")
# you deseasonalise it!
ds <- function(x) ...
plot(ds(o), type="l")
```

be lazy

use (i)apply

What does (i)apply do?

```
iapply(de$ozone, ?, length)
iapply(de$ozone, ?, mean)
iapply(de$ozone, ?, ds)
```

and don’t forget to make a function

```
iapply(de$ozone, 3, ...)
```

```
de$ozone[,,1]
de$ozone[,,2]
de$ozone[,,3]
de$ozone[,,4]
de$ozone[,,5]
```

```
iapply(de$ozone, 2, ...)
```

```
de$ozone[,1,]
de$ozone[,2,]
de$ozone[,3,]
de$ozone[,4,]
de$ozone[,5,]
```

```
iapply(de$ozone, 2:3, ...)

de$ozone[,1,1]
de$ozone[,1,2]
..
de$ozone[,2,1]
de$ozone[,2,2]
```

```
iapply(de$ozone, c(1,3), ...)

de$ozone[1,,1]
de$ozone[1,,2]
..
de$ozone[2,,1]
de$ozone[2,,2]
```

```
iapply(de$ozone, 1:3, ..)

de$ozone[1,1,1]
de$ozone[1,1,2]
..
de$ozone[2,1,1]
...
de$ozone[2,2,2]
```


Why not use a loop?

```
ds_all <- function(x)
   iapply(x, c(1,2), ds)
)
```

be lazy

use lapply

What does lapply do?

```
lapply(de, ds_all)
#it's that easy!
```

Common themes

- Start simple
- Build up functions as you go
- Use apply and friends to reduce the amount of code you have to write