# Have Env – Will Travel

Colin A. Gross
AARUG 2018-09-13

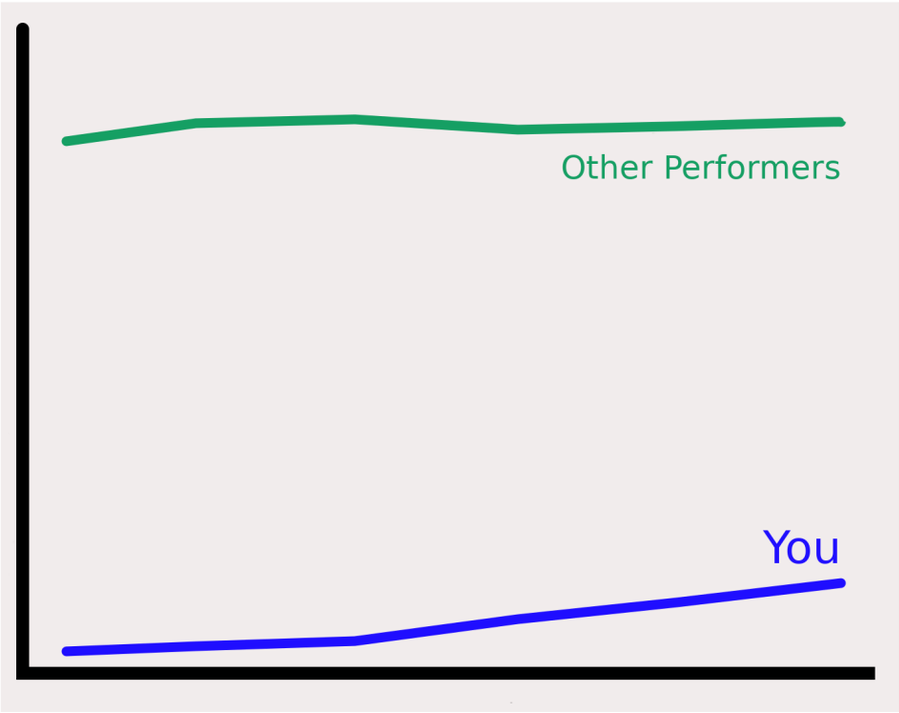
# Tailoring Performance Feedback

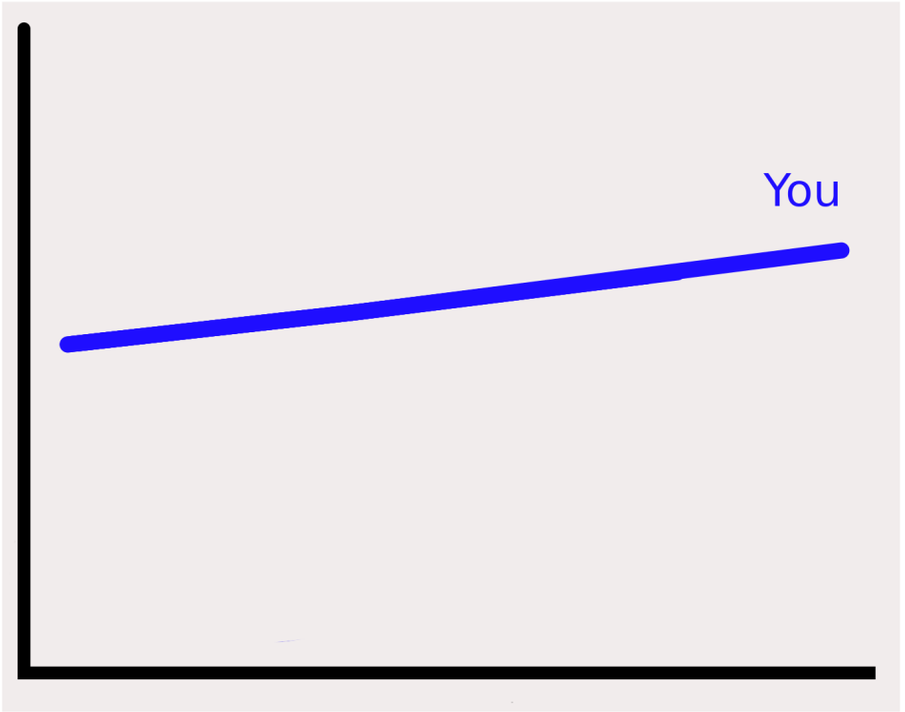
**One report does not fit all performers.**

**Psychology has theories about what works and when.**

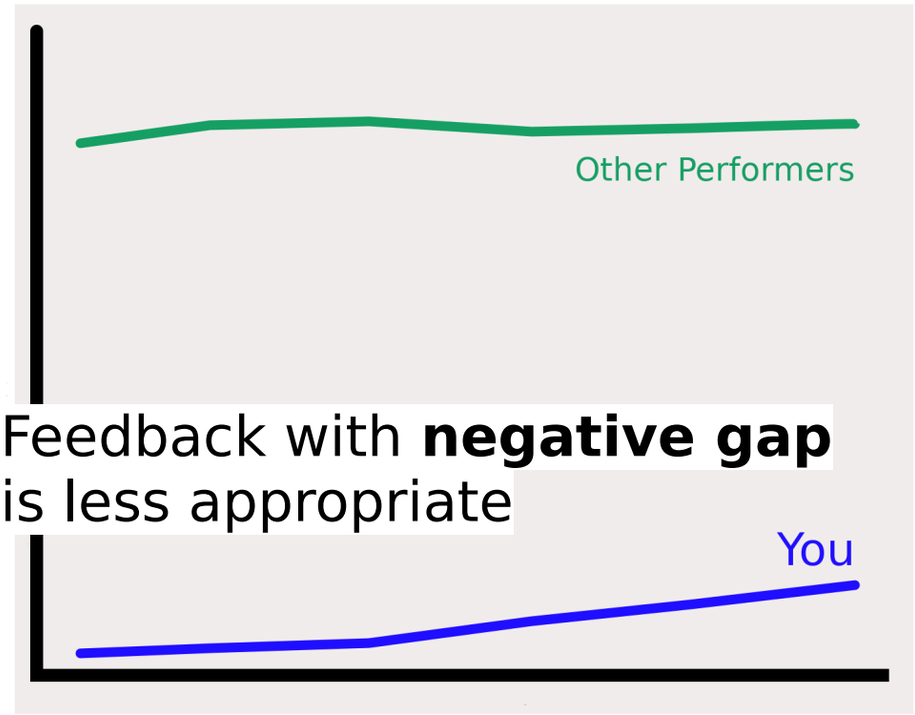
**Can analyze performance data to select better feedback.**

# E.g Feedback for a beginner

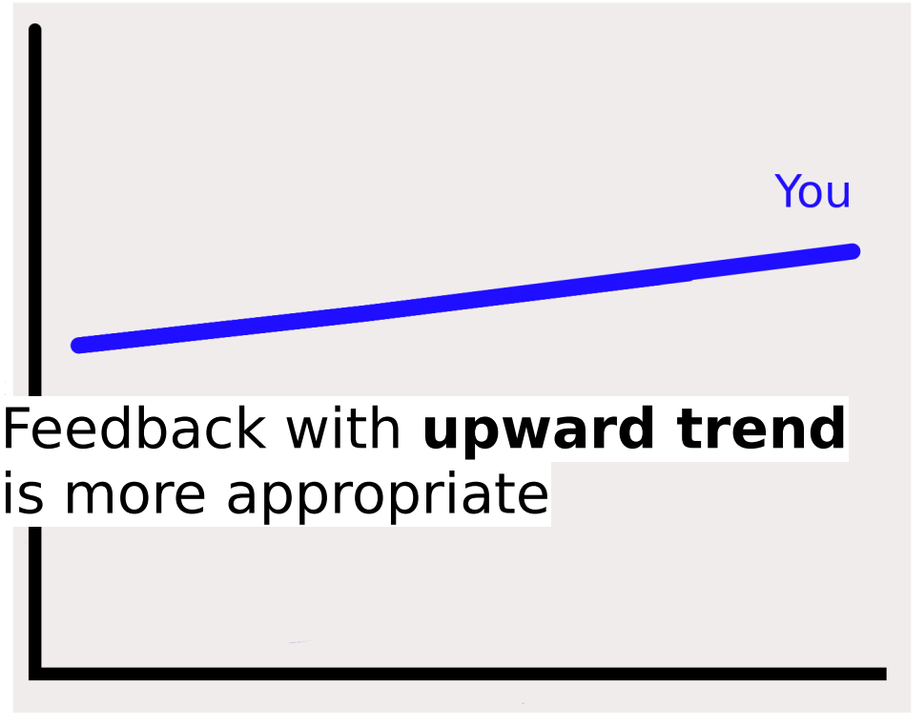


# E.g Feedback for a beginner




Feedback with **negative gap** is less appropriate




Feedback with **upward trend** is more appropriate

# Performer Attributes

**Tailoring depends on attributes like:**

**- negative_gap**

**- upward_trend**

**Specific to and specified by client**

*“An upward trend is when someone’s last three counts are increasing.”*

*“A negative gap is when the last count is less than 90% of the average.”*

# Example Data

| id | count |
| --- | --- |
| Alice | 4 |
| Bob | 6 |
| Carol | 8 |

# Getting Attributes from Data

*“A negative gap is when the last count is less than 90% of the average.”*

```
eval_gap(count, mean){
  ifelse( count < mean * 0.9, TRUE, FALSE ) }


annotate_perf_gap_neg <- function(data){
  data %>%
   mutate(mean = mean( count ),
          perf_gap_neg = eval_gap(count, mean)) %>%
   select(id, perf_gap_neg) }
```

# Example Annotation

| id | perf_gap_neg |
|---|---|
| Alice | TRUE |
| Bob | FALSE |
| Carol | FALSE |

# Handling Annotations

**Where to keep the functions.**

**A naming convention for annotations.**

**Standard parameters.**

**Reading in annotation functions.**

**A reasonable way to execute the functions.**

# Where to Keep Annotations

Keep along side the data and config for a client project.

```
/path/to/clients/
└── baz-clinic
    └── 2018-anti-microbial-stewardship
        ├── annotations.r
        ├── config.json
        └── performance-data.csv
```

# Annotation Interface

**Function naming convention:**

```
annotate_attribute_label
```

e.g. `annotate_upward_trend`

**Standard parameters:**

```
function(performance_data, column_specification)
```

# Sourcing Annotations

Source annotations into a new environment:

```
source_annotations <- function(path) {
  anno_env <- new.env(parent = .BaseNamespaceEnv)
  source(path, local = anno_env)
  return(anno_env)
}
```

local TRUE, FALSE or **an environment, determining where the parsed expressions are evaluated.**

# Executing Annotations

**Get a list of the functions.**

**Send the data to each of them.**

**Compile results.**

# Get List of Functions

Environment is a object on which we can run:

```
anno_func_names <-
    lsf.str(envir = anno_env, pattern = "annotate")
```

List Objects and their Structure

lsf.str return an object of class "ls_str", basically the **character vector of matching names (functions only for lsf.str**)

# Pass Data to Functions

```
# Build argument list to pass to each of the
annotation functions.
aargs <- list(data = data, col_spec = col_spec)

# Collect table per each annotation function
 anno_results <-
   lapply(anno_func_names,
          do.call,
          args = anno_args,
          envir = anno_env)
```

# Compile Results

```
# Reduce results list into a single table
Reduce(left_join, anno_results)
```

| id | perf_gap_neg | upward_trend | capability_barrier |
|---|---|---|---|
| Alice | TRUE | TRUE | TRUE |
| Bob | FALSE | TRUE | FALSE |
| Carol | FALSE | FALSE | FALSE |

# Nice Things About Environments

- Don’t pollute RGlobal_Env namespace.
- Can be passed around as a single object.
- Can be mocked for testing.

# Building Envs for Testing

```
bad_env < - new.env()
bad_env$foo < - function(){}


test_that('Annotate complains about no annotation functions.', {
  expect_success(expect_warning(
    annotate(mock_data, bad_env, col_spec)
  ))
})
```

# Non-Polluting* Environment

```
env_foo< - new.env()
env_foo$bar < - function(){print(“baz”)}

bar()
#> Error in bar() : could not find function "bar"

env_foo$bar()
#> [1] "baz"
```

* Calls to library() or require() in annotations.r will modify search path

# Alternatives

- Build packages of annotations
- Source directly into Rglobal_Env
- Have a separate branch in git for every client
- “How about a web page app that writes the settings that describes the math for the annotations?”

# End of Line