Understanding the R-Instat Calculation System

The R-Instat Calculation System

Learning Objectives

By the end of this page, you will be able to:

- Understand what an `instat_calculation` object is.
- Distinguish between different calculation types.
- Build multi-step calculations using manipulations and sub-calculations.
- Execute and reuse complex workflows in a `data_book`.

1. What is an `instat_calculation`?

In R-Instat, we use the `instat_calculation` class (from the `instatCalculations` package) to define calculations in a structured way. This allows for:

- Modular calculation pipelines
- Reproducible outputs
- Integration with metadata and links

Each calculation represents a step like filtering, mutating, summarising, or grouping — similar to how you'd use `dplyr`.

2. Anatomy of a Calculation

Each `instat_calculation` object has key fields:

Field	Description
`type`	What kind of step it is (e.g. "filter", "summary"), see section 3
`function_exp`	R code (as a string) to be evaluated in that step.
`result_name`	Name of the output column (for mutate or summarise).
`calculated_from`	What variables this step depends on.
`manipulations`	Earlier steps that transform the input before this one runs.
`sub_calculations`	Supporting steps that are required before the main step.
`save`	Controls what is saved: 0 = nothing, 1 = calculation only, 2 = save result too.

3. Types of Calculations

Type	Equivalent Function	Purpose
`"calculation"`	`mutate()`	Create new columns
`"filter"`	`filter()`	Select rows
`"summary"`	`summarise()`	Create summaries
`"by"`	`group_by()`	Group before summarising
`"sort"`	`arrange()`	Sort rows
`"combination"`	*(meta-type)*	Bundle multiple steps together

4. Example: Detecting the Start of the Rains

We'll use rainfall data and apply a set of transformations to calculate when the rainy season starts. Here, we want to calculate the first instance each year where there is more than 20mm of rain in 2 days.

Step 1: Create a Rolling Sum of the Rainfall

This is to calculate the rainfall over each 2 day period. We do not want to save this as a new column to our data as this is just an interim calculation.

```
roll_sum_rain <- instat_calculation$new(
  type = "calculation",
  function_exp = "RcppRoll::roll_sumr(x=rain, n=2, fill=NA, na.rm=FALSE)",
  result_name = "roll_sum_rain",
  calculated_from = list("dodoma" = "rain")
)
```

Step 2: Filter to where each rolling sum is >= 20

This then just gives us where the rainfall is over 20mm over a two day period. This is to calculate the rainfall over each 2 day period. We do not want to save this as a new column to our data as this is just an interim calculation. So we do not set the `save` parameter in here.

```
conditions_filter <- instat_calculation$new(
  type = "filter",
  function_exp = "((rain >= 0.85) & roll_sum_rain > 20) | is.na(x=rain) | is.na(x=roll_sum_rain)",
  sub_calculations = list(roll_sum_rain)
)
```

Step 3: Group by each year

We want this to be for each year, since we want to get the first instance of this for each year.

```
grouping_by_year <- instat_calculation$new(
  type = "by",
  calculated_from = list("dodoma" = "year")
)
```

Step 4: Filter by Day of Year (DOY)

Let's say we just wanted to look from 31st March (day 91) until mid-May (day 136)

```
doy_filter <- instat_calculation$new(
  type = "filter",
  function_exp = "doy_366 >= 91 & doy_366 <= 136",
  calculated_from = list("dodoma" = "doy_366")
)
```

Step 5: Get the Value!

Summarise to get the start day

```
start_of_rains_doy <- instat_calculation$new(
  type = "summary",
  function_exp = "ifelse(is.na(first(rain)) | is.na(first(roll_sum_rain)), NA, first(doy_366, default=NA
  result_name = "start_rain",
  save = 2
)
```

Step 6: Combine the Steps

We want to combine the steps next to tell it to run everything

```
start_of_rains_combined <- instat_calculation$new(
  type = "combination",
  manipulations = list(conditions_filter, grouping_by_year, doy_filter),
  sub_calculation = list(start_of_rains_doy)
)
```

Step 7: Run the Calculation

Now we tell the calculation to run

```
data_book$run_instat_calculation(
  calc = start_of_rains_combined,
  display = FALSE,
  param_list = list(drop = FALSE)
)
```

The output variable `start_rain` will be added to the appropriate linked data frame.

Best Practices

- Use `"combination"` to wrap multi-step logic.
- Use `save = 2` only for results you want to keep in the dataset.
- Keep `calculated_from` updated to maintain traceability.
- You can remove all intermediate calculation objects with `rm()` after use.

5. Task to Try

Create a summary calculation to compute the **mean rainfall per year**, using:

- a `group_by` step
- a `summary` step
- a `combination` to wrap them
- `save = 2` to store the result in the linked dataset

6. Conclusion

The R-Instat calculation system offers a powerful, modular way to define and run complex data transformations – all while preserving structure, reproducibility, and integration with linked datasets.

By using `instat_calculation` objects, you can:

- Chain multiple steps into logical workflows
- Keep calculations reusable and consistent
- Work with complex datasets without altering the original structure
- Generate new variables and summaries that respect grouping, filtering, and metadata