

kscript

Scripting Enhancements for Kotlin

Holger Brandl

3.11.2017 - KotlinConf2017

kscript - Scripting Enhancements for Kotlin



Holger Brandl

[@holgerbrandl](https://twitter.com/holgerbrandl)

www.github.com/holgerbrandl

What are language requirements for data-driven science?

Why Kotlin may be the right choice.

How `kscript` provides enhanced scripting support for Kotlin.

Language requirements for data-driven science

1. Reproducible: Can we reproduce results at a later date?
2. Rapid Prototyping: Can we quickly setup an analysis?
3. Scalable: Can we upscale scripts into systems and tools?
4. Community: Can we enable colleagues to analyze data?

Use just the linux shell for data-driven science?

■ But it's faster and way shorter to do it bash! (unknown colleague)

Do *magical* things in the terminal without dependencies

Use just the linux shell for data-driven science?

But it's faster and way shorter to do it bash! (unknown colleague)

Do *magical* things in the terminal without dependencies

```
removeMultiMappers() {  
perl -ne 'unless((!($_=~/^@.*/)&&!(m/AS:i:/)) || (m/AS:i:(\S+).+XS:i:(\S+)/ && $1-$2<1)){print;}'  
}  
  
bowtie2 $fastqFile | samtools view -SF 4 - | removeMultiMappers | samtools sort - result.bam
```

Use just the linux shell for data-driven science?

But it's faster and way shorter to do it bash! (unknown colleague)

Do *magical* things in the terminal without dependencies

```
removeMultiMappers(){  
perl -ne 'unless((!($_=~/^@.*/)&&!(m/AS:i:/))||(m/AS:i:(\S+).+XS:i:(\S+)/ && $1-$2<1)){print;}'  
}  
  
bowtie2 $fastqFile | samtools view -SF 4 - | removeMultiMappers | samtools sort - result.bam
```

Marta Florio, Mareike Albert, Elena Taverna, Takashi Namba, **Holger Brandl**, Eric Lewitus, Christiane Haffner, Alex Sykes, Fong Kuan Wong, Julia Peters, E. Guhr, Sylvia Klemroth, Kay Prüfer, Janet Kelso, Ronald Naumann, Ina Nüsslein, Andreas Dahl, Robert Lachmann, Svante Pääbo, Wieland B. Huttner **Human-specific gene ARHGAP11B promotes basal progenitor amplification and neocortex expansion.** *Science* , 347(6229) 1465-1470 (2015)

Use just the linux shell for data-driven science?

But it's faster and way shorter to do it bash! (unknown colleague)

Do *magical* things in the terminal without dependencies

```
removeMultiMappers(){  
perl -ne 'unless(!(($_=~/^@.*/)&&!(m/AS:i:/))||(m/AS:i:(\S+).+XS:i:(\S+)/ && $1-$2<1)){print;}'  
}  
  
bowtie2 $fastqFile | samtools view -SF 4 - | removeMultiMappers | samtools sort - result.bam
```

Marta Florio, Mareike Albert, Elena Taverna, Takashi Namba, **Holger Brandl**, Eric Lewitus, Christiane Haffner, Alex Sykes, Fong Kuan Wong, Julia Peters, E. Guhr, Sylvia Klemroth, Kay Prüfer, Janet Kelso, Ronald Naumann, Ina Nüsslein, Andreas Dahl, Robert Lachmann, Svante Pääbo, Wieland B. Huttner **Human-specific gene ARHGAP11B promotes basal progenitor amplification and neocortex expansion.** *Science* , 347(6229) 1465-1470 (2015)

Get published and forget.

Fast to write. Impossible to maintain or to evolve. Does not scale. Unteachable black art.

Java for Data Science?

2001 - First contact:

- ▮ **Evolving image processing for 2d gel electrophoresis analysis**

Java for Data Science?

2001 - First contact:

Evolving image processing for 2d gel electrophoresis analysis

Likes

- Static typing provides guidance and context
- Type- and compile-time checking of code
- Ease of deployment
- Java dependency trees rock!

Java for Data Science?

2001 - First contact:

Evolving image processing for 2d gel electrophoresis analysis

Likes

- Static typing provides guidance and context
- Type- and compile-time checking of code
- Ease of deployment
- Java dependency trees rock!

Dislikes

- Prototyping in the debugger
- Rerun apps again and again
- Lack of interactivity

Java for Data Science?

2001 - First contact:

Evolving image processing for 2d gel electrophoresis analysis

Likes

- Static typing provides guidance and context
- Type- and compile-time checking of code
- Ease of deployment
- Java dependency trees rock!

Dislikes

- Prototyping in the debugger
- Rerun apps again and again
- Lack of interactivity

Java allows to write complex software but fails for explorative analysis and prototyping.

Java for Data Science?

1. Reproducible: Can we reproduce results at a later date? ✓
2. Rapid Prototyping: Can we quickly setup an analysis? ✗
3. Scalable: Can we upscale scripts into systems and tools? ✓
4. Community: Can we enable colleagues to analyze data? ✗

Most popular choices for data science?

R stronger for statistics and explorative data-analysis (dplyr, tidyr, rstudio).

Python more popular for machine learning (kaggle, keras, scikit)

Both are fun to teach and great for prototyping.



Python Dependencies: The *best* practice

```
virtualenv my_scikit  
  
source my_scikit/bin/activate  
  
pip install -U scikit-learn  
  
## actual logic goes in here!  
/sw/bin/python -c '  
from sklearn import datasets  
... the cool stuff!  
'  
  
deactivate
```

- Per project index only via `virtualenv`, which is tedious for small scale tasks
- No binaries, always (slowly) compiles from scratch

No sound way to manage dependencies in python

R dependencies

- No built-in way to indicate a specific dependency version for a project
- Also declaration of package-dependencies is sloppy by design

```
Imports: assertthat, bindrcpp (>= 0.2), glue (>= 1.1.1), magrittr,  
         methods, pkgconfig, rlang (>= 0.1.2), R6, Rcpp (>= 0.12.7),  
         tibble (>= 1.3.1), utils
```

- Existing approaches like **packrat** are not yet ready for prime time: build package index from scratch per project

R dependencies

- No built-in way to indicate a specific dependency version for a project
- Also declaration of package-dependencies is sloppy by design

```
Imports: assertthat, bindrcpp (>= 0.2), glue (>= 1.1.1), magrittr,  
         methods, pkgconfig, rlang (>= 0.1.2), R6, Rcpp (>= 0.12.7),  
         tibble (>= 1.3.1), utils
```

- Existing approaches like **packrat** are not yet ready for prime time: build package index from scratch per project

Python and R fail to be *the* solution for data science

1. Reproducible: Can we reproduce results at a later date? ❌
2. Rapid Prototyping: Can we quickly do setup an analysis? ✅
3. Scalable: Can we upscale scripts into systems and tools? ❌
4. Learning curve: Can we enable colleagues to analyze data? ✅

How to deploy software for data-driven science?

Webservices don't scale with data. Bring tools to users. Example:

Raphael Etournay, Marko Popović, Matthias Merkel, Amitabha Nandi, Corinna Blasse, Benoit Aigouy, Holger Brandl, Gene Myers, Guillaume Salbreux, Frank Jülicher, Suzanne Eaton
Interplay of cell dynamics and epithelial tension during morphogenesis of the *Drosophila* pupal wing.
Elife , 4 Art. No. e07090 (2015)

The background is a dark, textured surface, possibly a piece of leather or a similar material, with a fine, irregular pattern. Overlaid on this is a large, abstract shape composed of two distinct colors: a bright yellow and a vibrant green. The yellow shape is on the left, and the green shape is on the right, with some overlap between them. The text "How to ship it to the community?" is centered over the yellow and green shapes in a white, sans-serif font.

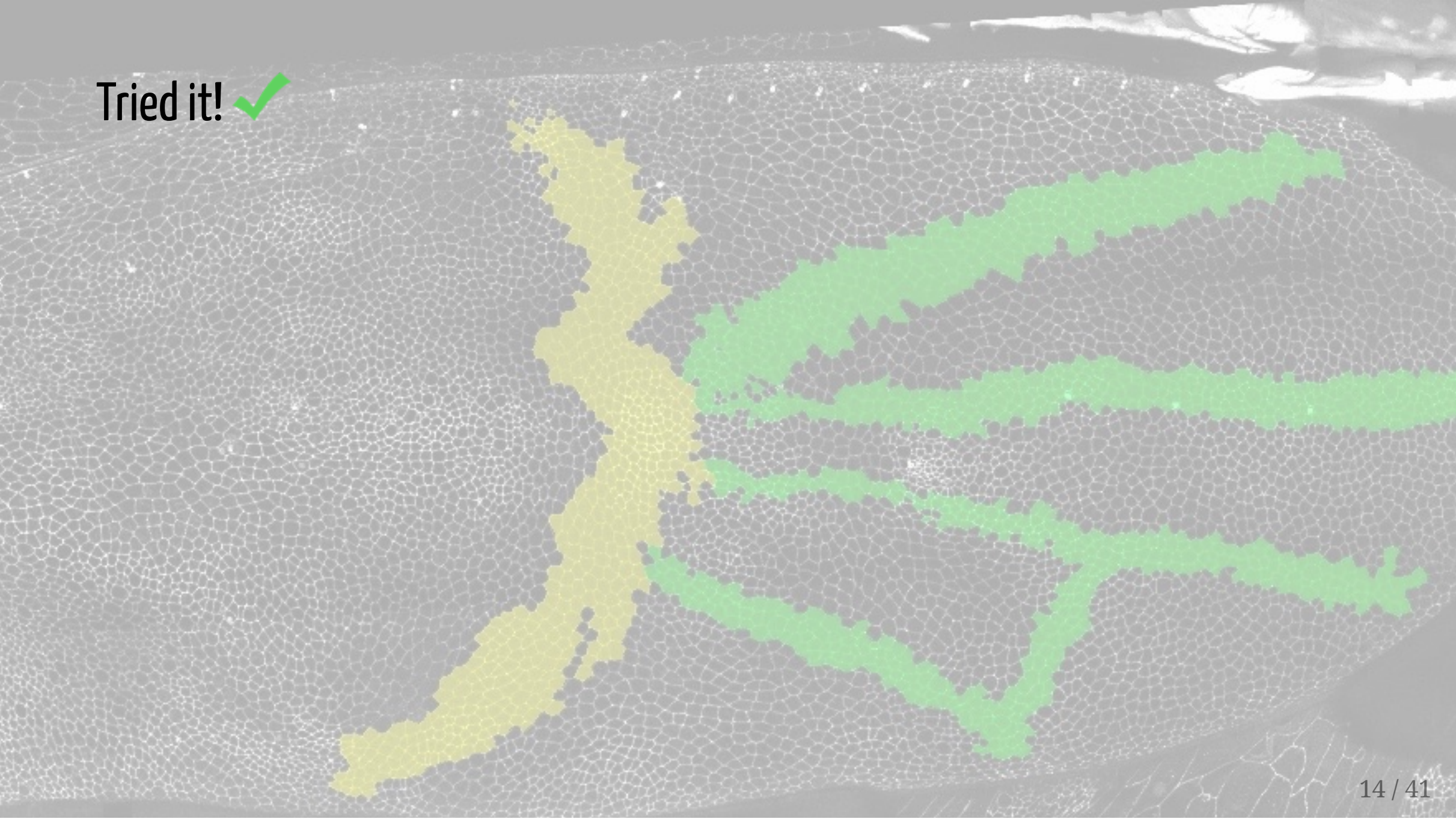
How to ship it to the community?

A close-up photograph of a black, textured surface, likely a car seat or dashboard, with a fine, pebbled grain. Overlaid on this background is a stylized graphic consisting of a yellow, jagged, vertical shape on the left and several green, horizontal, wavy lines extending from the yellow shape towards the right. The text is centered over the yellow and green areas.

How to ship it to the community?

Dockerize it!

Tried it! ✓





Tried it! ✓

Sucessfully published! ✓

Raphael Etournay, Matthias Merkel, Marko Popović, **Holger Brandl**, Natalie Dye, Benoit Aigouy, Guillaume Salbreux, Suzanne Eaton, Frank Jülicher **TissueMiner: a multiscale analysis toolkit to quantify how cellular processes create tissue dynamics**. *Elife* , 5 Art. No. e14334 (2016)

https://github.com/mpicbg-scicomp/tissue_miner

Tried it! ✓

Successfully published! ✓

Raphael Etournay, Matthias Merkel, Marko Popović, **Holger Brandl**, Natalie Dye, Benoit Aigouy, Guillaume Salbreux, Suzanne Eaton, Frank Jülicher **TissueMiner: a multiscale analysis toolkit to quantify how cellular processes create tissue dynamics**. *Elife*, 5 Art. No. e14334 (2016)

https://github.com/mpicbg-scicomp/tissue_miner

```
alias tm='docker run --rm -ti -v $(dirname $PWD):/movies etournay/tissue_miner'

tm analyze_sheer <image_data>
tm track_cell_divisions <image_data>
```

Tried it! ✓

Successfully published! ✓

Raphael Etournay, Matthias Merkel, Marko Popović, **Holger Brandl**, Natalie Dye, Benoit Aigouy, Guillaume Salbreux, Suzanne Eaton, Frank Jülicher **TissueMiner: a multiscale analysis toolkit to quantify how cellular processes create tissue dynamics**. *Elife* , 5 Art. No. e14334 (2016)

https://github.com/mpicbg-scicomp/tissue_miner

```
alias tm='docker run --rm -ti -v $(dirname $PWD):/movies etournay/tissue_miner'

tm analyze_sheer <image_data>
tm track_cell_divisions <image_data>
```

Doing *just* docker-support since then. ✗

A better way to do data science?

- Reproducibility, Dependencies & Scalability: **Java**
- Prototyping & Community: **R & Python**

Fusion would be great:

Java with the ease of a scripting language

A better way to do data science?

- Reproducibility, Dependencies & Scalability: **Java**
- Prototyping & Community: **R & Python**

eierlegende Wollmilchsau

Fusion would be great:

Java with the ease of a scripting language

February 2016

Kotlin v1.0 released

Type Inference

Extension Functions

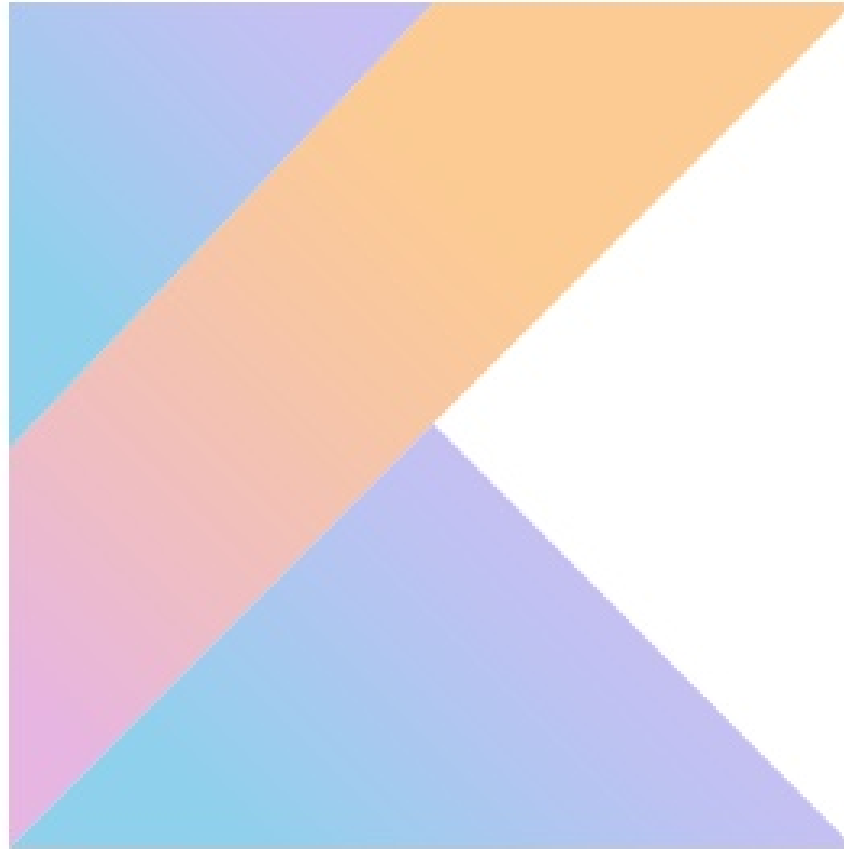
Data Classes

DSLs in Mind

Default Parameters

Lives in JVM

Scripting Support



February 2016

Kotlin v1.0 released

Type Inference

Extension Functions

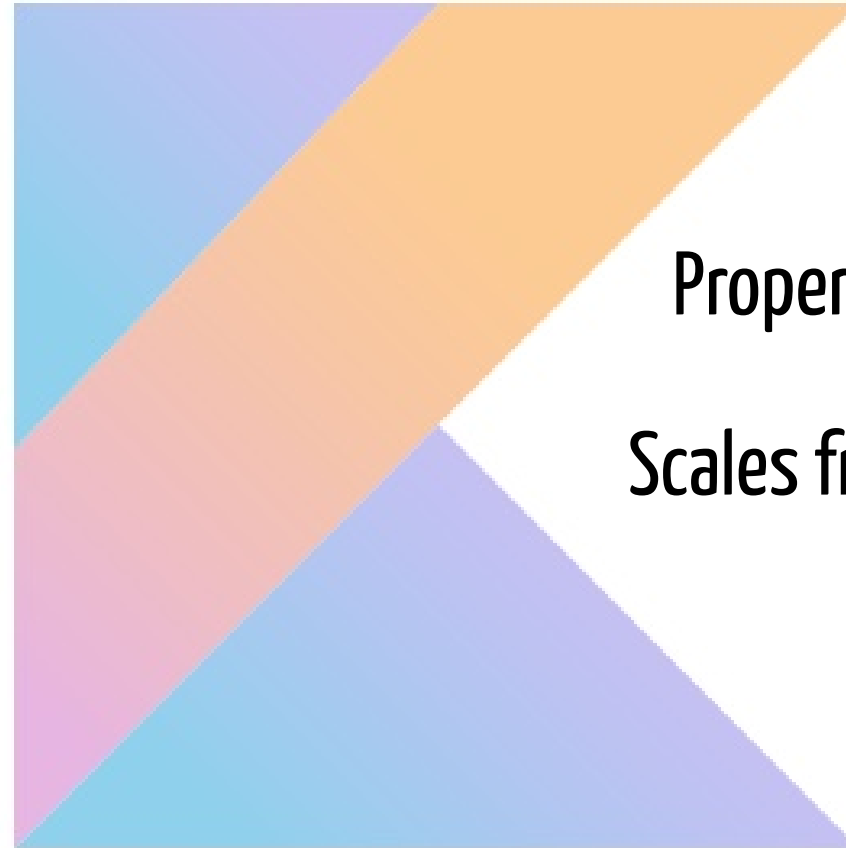
Data Classes

DSLs in Mind

Default Parameters

Lives in JVM

Scripting Support



Data Science Requirements

Proper Dependency Model ✓

Scales from idea to product ✓

Teachable ✓

Rapid Prototyping ??

Let's write a script!

```
kotlinc -script pipeline_prototype.kts
```

which is equivalent to

```
#!/usr/bin/env kotlinc -script  
  
println("do stuff")  
  
val data = readData() // not quite yet!
```

Issues with existing out-of-the-box `kotlinc` tooling

- How to pull in dependencies? `kotlinc -cp ...` seems too tedious. A separate gradle/maven file even more.
- `kotlinc` means recompile, so little scriptlets run slow
- How to keep track of required JVM/runtime options?
- More flexible way to provide scripts

holgerbrandl / kscript

Unwatch 9 Unstar 157 Fork 5

Code Issues 6 Pull requests 1 Projects 0 Wiki Insights Settings

Scripting enhancements for Kotlin Edit

kotlin bash scripting dependency-resolution mpi-cbg Manage topics

229 commits 3 branches 9 releases 2 contributors MIT

Branch: master New pull request Create new file Upload files Find file Clone or download

Since March 2016

Enhanced scripting support for **Kotlin** on *nix-based systems.

kscript provides an easy-to-use, very flexible, and almost zero-overhead solution to write self-contained mini-applications with Kotlin.

Core features of `kscript`

Wraps `kotlin` and adds

Compiled script caching (using md5 checksums)

Dependency declarations using gradle-style resource locators

More options to provide scripts

Embedded runtime options

Support library to ease the writing of Kotlin scriptlets

```
# sdk install kscript ## for details see http://sdkman.io/  
kscript --help
```

kscript - Enhanced scripting support for Kotlin on *nix-based systems.

Usage:

```
kscript [options] <script> [<script_args>]...
```

```
kscript --clear-cache
```

```
kscript --self-update
```

The <script> can be a script file (*.kts), a script URL, - for stdin, a *.kt source file with a main method, or some Kotlin code.

Use '--clear-cache' to wipe cached script jars and urls

Use '--self-update' to wipe cached script jars and urls

Options:

-i --interactive	Create interactive shell with dependencies as declared in script
-t --text	Enable stdin support API for more streamlined text processing
--idea	Open script in temporary IntelliJ session

Copyright : 2017 Holger Brandl

License : MIT

Version : v2.1.0

Website : <https://github.com/holgerbrandl/kscript>

Input Modes

There's a multitude of ways to serve a script to `kscript`

The `<script>` argument can be a

- script file (`*kts`),
- a script URL,
- `-` for stdin,
- some kotlin code
- a `*.kt` source file with a main method,

Interpreter Usage

To use `kscript` as interpreter for a script `Example.kts` just point to it in the shebang line

```
#!/usr/bin/env kscript  
println("hello world")
```

We can still run it directly with

```
kscript Example.kts
```

Or make executable and run directly

```
chmod +x Example.kts  
./Example.kts
```

Inlined Usage

Use `kscript` in a workflow without creating an additional script file.

- Directly provide a Kotlin scriptlet as argument

```
kscript 'println("hello world")'
```

- Pipe a Kotlin snippet into `kscript` and instruct it to read from `stdin` by using `-` as script argument

```
echo '
println("Hello Kotlin.")
' | kscript -
```

- Use heredoc (preferred solution for inlining) which gives more flexibility. E.g. use single quotes in script

```
kscript - <<"EOF"
println("It's a beautiful day!")
EOF
```

kscript can launch apps right from the internet

Example:

```
kscript 'https://git.io/v9R73' arg u ments
```

To streamline the usage, the first part could be even aliased:

```
alias hello_kscript="kscript https://git.io/v1cG6"  
hello_kscript my argu ments
```

- Support remote tool repositories
- Allows for easy integration of remotely hosted (mini) programs into data pipelines
- Installation-free
- URLs are assumed to be static and will be cached via checksum.

Write command line tools with *just* the standard library

```
#!/usr/bin/env kscript

val stdin: Sequence<String> = generateSequence { readLine() }

stdin
    // retain only lines starting with digits 0 to 5
    .filter{ it.contains("[0-5]{5}".toRegex()) }
    // add prefix
    .map { "id" + it }
    // print to stdout
    .forEach { println(it) }
```

Would work using `kotlinc -script` as interpreter as well, but would be painfully slow

Be snappy by caching compiled scripts

`kscript` execution model

1. Calculate md5 of input script
2. Look up cached jar by md5; if not yet present create it
3. Use process substitution to replace `kscript` with `kotlin` process

All *kscripts* are cached based on md5 checksum, so running the same snippet again will be much faster

The cache directory is `~/.kscript` and can be cleared with `kscript --clear-cache`

Directives

Script sources are (often) not enough to ... *write self-contained mini-applications with Kotlin*. (from kscript README)

Directives supported by kscript:

//DEPS	Declare dependencies with gradle-style locators
//KOTLIN_OPTS	Configure the kotlin/java runtime environment
//INCLUDE	Source kotlin files into the script
//ENTRY	Declare application entrypoint for kotlin <code>*.kt</code> applications

Declare dependencies with //DEPS

```
#!/usr/bin/env kscript
```

```
//DEPS com.beust:klaxon:0.24, com.github.kittinunf.fuel:fuel:1.3.1
```

```
import com.beust.klaxon.*
```

```
import com.github.kittinunf.fuel.httpGet
```

```
require(args.isNotEmpty()) {  
    println("Usage: id_converter <some_id>+")  
    kotlin.system.exitProcess(-1)  
}
```

```
val queryURL = "http://foo.com/bar?convert=${args.joinToString(",")}"
```

```
val json = String(queryURL.httpGet().response().second.data)
```

```
val jsonArray = Parser()  
    .parse(json.byteInputStream())!! as JsonArray<*>
```

```
// use klaxon library to parse the json result
```

```
val conversionTable = jsonArray.map { (it as JsonObject) }.map {  
    it.int("id") to it.string("converted")  
}.forEach { println(it) }
```

Configure the runtime with `//KOTLIN_OPTS`

Just passed on to `kotlin`.

Example: Filter dna-sequences by length

```
#!/usr/bin/env kscript

//DEPS de.mpicbg.scicomp:kutils:0.4
//KOTLIN_OPTS -J-Xmx5g -J-server

import de.mpicbg.scicomp.bioinfo.openFasta

if (args.size != 2) {
    System.err.println("Usage: fasta_filter <fasta> <max_length>")
    kotlin.system.exitProcess(-1)
}

val fastaFile = java.io.File(args[0])
val lengthCutoff = args[1].toInt()

openFasta(fastaFile).
    filter { it.sequence.length <= lengthCutoff }.
    forEach { print(it.toEntryString()) }
```

Ease prototyping with `//INCLUDE`

Many other languages allow to directly include source files. Can we do the same in Kotlin? Yes, with `kscript`!

```
//utils.kt
fun Array<Double>.median(): Double {
    val (lower, upper) = sorted().let { take(size / 2) to takeLast(size / 2) }
    return if (size % 2 == 0) (lower.last() + upper.first()) / 2.0 else upper.first()
}
```

Artifacts are the proper way!

`kscript` allows to skip artifact deployment by using `//INCLUDE`

```
#!/usr/bin/env kscript
```

```
//INCLUDE utils.kt
//INCLUDE https://github.com/krangl/blob/src/MathHelpers.kt
```

```
val robustMean = listOf(1.3, 42.3, 7.).median()
println(robustMean)
```

Use `//ENTRY` to run applications with `main` method

`kscript` also supports running regular Kotlin `kt` files.

Example: `./examples/Foo.kt`:

```
package examples

//ENTRY examples.Bar

class Bar{
    companion object {
        @JvmStatic fun main(args: Array<String>) {
            println("Foo was called")
        }
    }
}

fun main(args: Array<String>) = println("main was called")
```

To run top-level `main` instead we would use `//ENTRY examples.FooKt`

The latter is the default for `kt` files and could be omitted

Replace `awkward` terminal programming with Kotlin

`kscript` can be used as a speedier and more flexible substitute for built-in terminal tools such as `awk` or `sed`

Use `-t/--text` to enable a support API for line-based text processing

- Delete a column

```
awk '!(($3=""))' some_flights.tsv  
kscript -t 'lines.split().select(-3).print()' some_flights.tsv
```

- Delete trailing white space (spaces, tabs)

```
awk '{sub(/[ \t]*$/, "");print}' file.txt  
kscript -t 'lines.map { it.trim() }.print()' file.txt
```

See <https://github.com/holgerbrandl/kscript-support-api> and `kscript` as substitute for `awk`

Pimp your REPL with `--interactive`

```
#!/usr/bin/env kscript
//DEPS de.mpicbg.scicomp:kutils:0.4
import de.mpicbg.scicomp.bioinfo.openFasta

if (args.size != 1) {
    System.err.println("Usage: CountRecords <fasta>")
    kotlin.system.exitProcess(-1)
}

val records = openFasta(java.io.File(args[0]))
println(records.count())
```

Pimp your REPL with `--interactive`

```
#!/usr/bin/env kscript
//DEPS de.mpicbg.scicomp:kutils:0.4
import de.mpicbg.scicomp.bioinfo.openFasta

if (args.size != 1) {
    System.err.println("Usage: CountRecords <fasta>")
    kotlin.system.exitProcess(-1)
}

val records = openFasta(java.io.File(args[0]))
println(records.count())
```

Bootstrap interactive shell from your script.

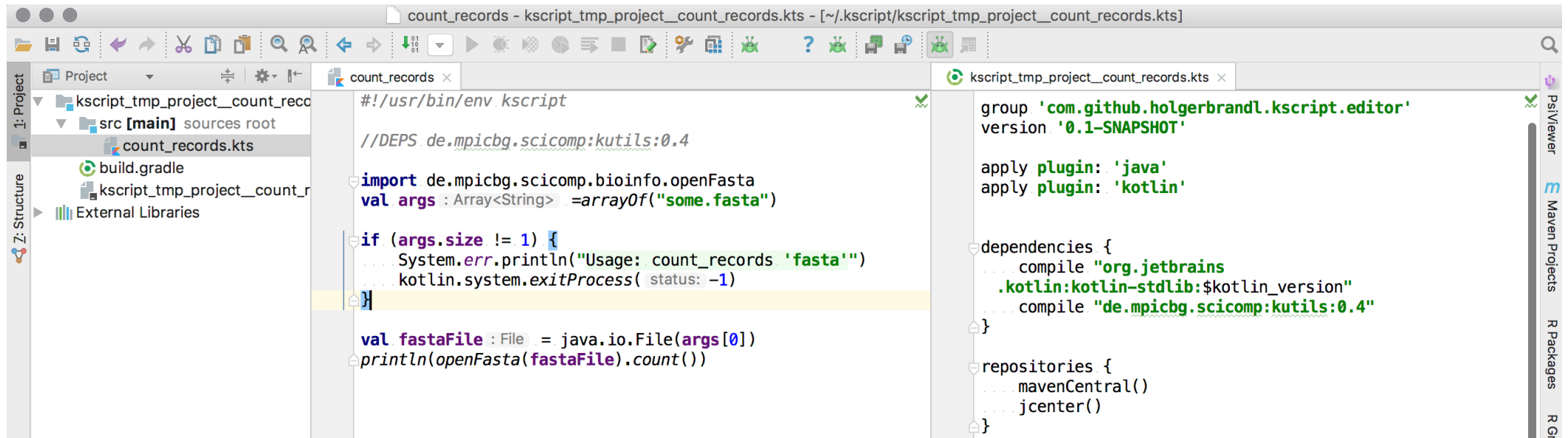
```
kscript --interactive CountRecords.kts
```

```
Creating REPL from /Users/brandl/Dropbox/kscript_kotlinconf_2017/code_examples/count_records.kts
Welcome to Kotlin version 1.1.51 (JRE 1.8.0_151-b12)
>>> import de.mpicbg.scicomp.bioinfo.openFasta
>>>
```

Bootstrap projects from scripts with `--idea`

- Artifacts and versions will differ between scripts
- `kscript` allows to create temporary project from script using Gradle

```
kscript --idea CountRecords.kts
```



Requires command-line launcher `idea` to be setup via Tools->Command Line Launcher

What's next?

Replace comment `//` directives with annotations

Using proper file annotations would be better design and also allow for better tooling

```
// DEPS log4j:log4j:1.2.14  
//vs  
@file:DependsOn("log4j:log4j:1.2.14")
```

Replace comment `//` directives with annotations

Using proper file annotations would be better design and also allow for better tooling

```
// DEPS log4j:log4j:1.2.14
//vs
@file:DependsOn("log4j:log4j:1.2.14")
```

No such thing as `DependsOn` in kotlin stdlib. To enable correct code parsing add `com.github.holgerbrandl:kscript-annotations:1.0` as dependency to the IntelliJ project

```
#!/usr/bin/env kscript

@file:DependsOn("de.mpicbg.scicomp:kutils:0.4")
@file:DependsOn("com.beust:klaxon:0.24", "com.github.kittinunf.fuel:fuel:1.3.1")

@file:KotlinOpts("-J-server") vararg args: String
@file:KotlinOpts("-J-Xmx5g")

@file:Include("util.kt")
@file:EntryPoint("Foo.bar") // applies on for kt-files

print("do something really cool here!")
```

Getting serious about scripting with KEEP #75

Great proposal that aims to unify various aspects of scripting in a common API

- Define Kotlin scripting and its applications
- Describe intended use cases for the Kotlin scripting
- Provide sufficient control of interpretation and execution of scripts
- Provide usable default components and configurations for the typical use cases

Getting serious about scripting with KEEP #75

Great proposal that aims to unify various aspects of scripting in a common API

- Define Kotlin scripting and its applications
- Describe intended use cases for the Kotlin scripting
- Provide sufficient control of interpretation and execution of scripts
- Provide usable default components and configurations for the typical use cases

Some core aspects are implemented by `kscript`!

- IDE/tooling support via `--idea`
- Project-level REPL `--interactive`
- Directives/annotations to declare dependencies and modulate the runtime environment

Getting serious about scripting with KEEP #75

Great proposal that aims to unify various aspects of scripting in a common API

- Define Kotlin scripting and it's applications
- Describe intended use cases for the Kotlin scripting
- Provide sufficient control of interpretation and execution of scripts
- Provide usable default components and configurations for the typical use cases

Some core aspects are implemented by `kscript`!

- IDE/tooling support via `--idea`
- Project-level REPL `--interactive`
- Directives/annotations to declare dependencies and modulate the runtime environment

Will it kill `kscript`? Maybe.

Looking forward to tool convergence!

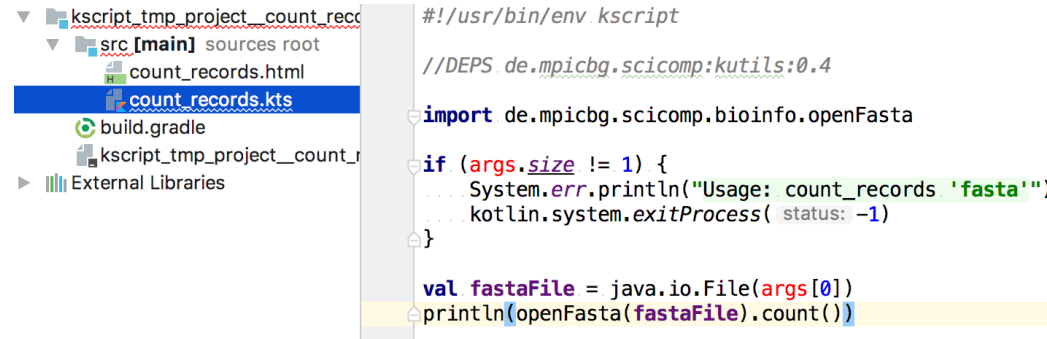


Next steps?

- Go kotlin-native to remove ~200ms JVM launch overhead

```
exec $(kotlin -classpath ${jarPath} kscript.app.KscriptKt "$@")
```

- Support for custom artifact repositories
- Pray for **KT-16802** *Good code is always red when editing kts-script files under Kotlin v1.1 to become fixed*



Not much left to do. :-) Don't get infected by *featureritis*!

kscript is unit- and battle-tested, and ready production!

Dream, dream, dream...

- Revise annotation-driven script configuration once **KEEP75** Scripting API supports it
- Use **kohesive/keplin-maven-resolver** for dependency resolution instead of mvn
- DSL-support: Derive new interpreters from kscript

```
#!/usr/bin/env tornando-fx-script

class HelloWorld : View() {
    override val root = hbox {
        label("Hello world")
    }
}
```

- *<your ideas here!>* just go to <https://github.com/holgerbrandl/kscript>

What's still missing in Kotlin to rule data-science?

1. **Interactivity:** Improve jupyter-kernel support (interactive usage, idea notebook editor support)
2. **More Kotlin DataScience APIs** for modeling (**pandas**, **dplyr**) and visualization (**ggplot2**, **seaborn**, **d3**)
3. **Fix the REPL** to be not just a re wihout pl
4. **Data-Science IDE** Provide a *cut down* version of IJ for data science

What's still missing in Kotlin to rule data-science?

1. **Interactivity:** Improve jupyter-kernel support (interactive usage, idea notebook editor support)
2. **More Kotlin DataScience APIs** for modeling (**pandas**, **dplyr**) and visualization (**ggplot2**, **seaborn**, **d3**)
3. **Fix the REPL** to be not just a re wihout pl
4. **Data-Science IDE** Provide a *cut down* version of IJ for data science

What's still missing in Kotlin to rule data-science?

1. **Interactivity:** Improve jupyter-kernel support (interactive usage, idea notebook editor support)
2. **More Kotlin DataScience APIs** for modeling (**pandas**, **dplyr**) and visualization (**ggplot2**, **seaborn**, **d3**)
3. **Fix the REPL** to be not just a re wihout pl
4. **Data-Science IDE** Provide a *cut down* version of IJ for data science

Summary

Kotlin may be the right choice for data science.

`kscript` provides enhanced scripting support for Kotlin.

`kscript` allows to replace scripts written in `ruby`, `python`, `bash`, `r` etc. with Kotlin

Summary

Kotlin may be the right choice for data science.

`kscript` provides enhanced scripting support for Kotlin.

`kscript` allows to replace scripts written in `ruby`, `python`, `bash`, `r` etc. with Kotlin

Questions and pull requests are welcome.

For these slides, more docs, examples see <https://github.com/holgerbrandl/kscript>

Thanks for your attention.

Thanks to the contributors of `kscript` for their great feedback and help.