Corpus-Supported Linguistic Research

Andres Karjus

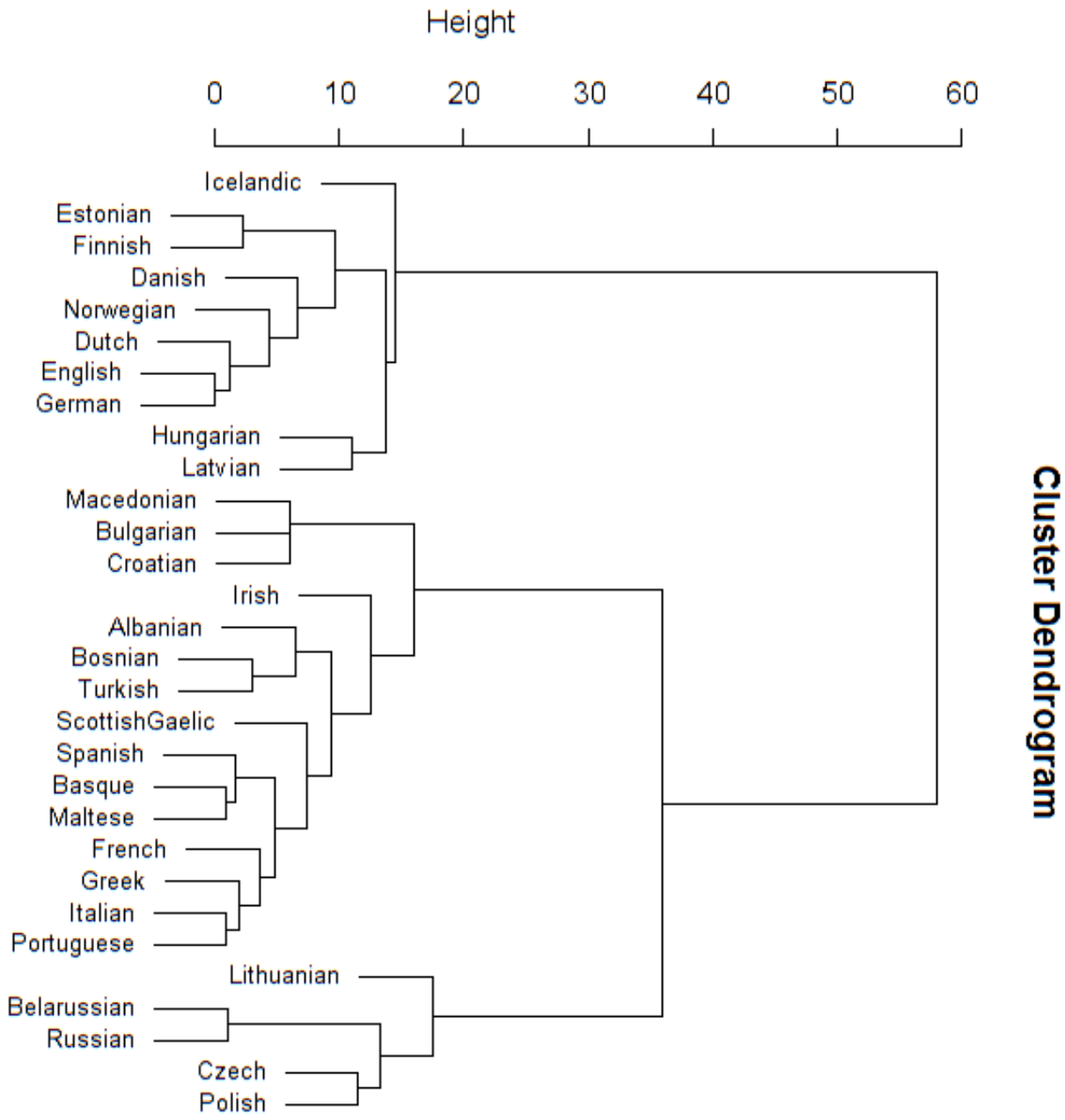
Academia Salensis, July 28 - August 1 2015

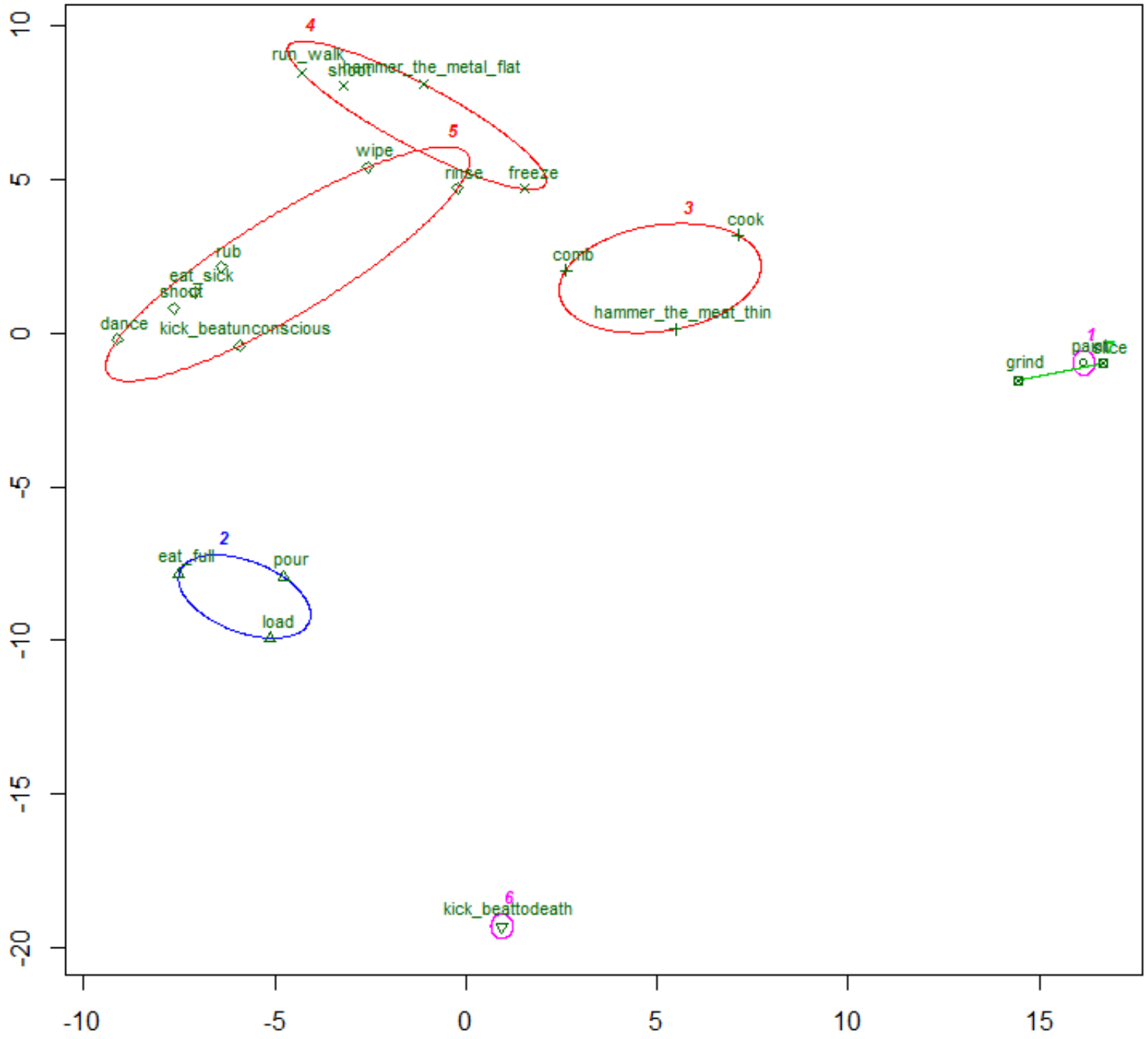
Focus

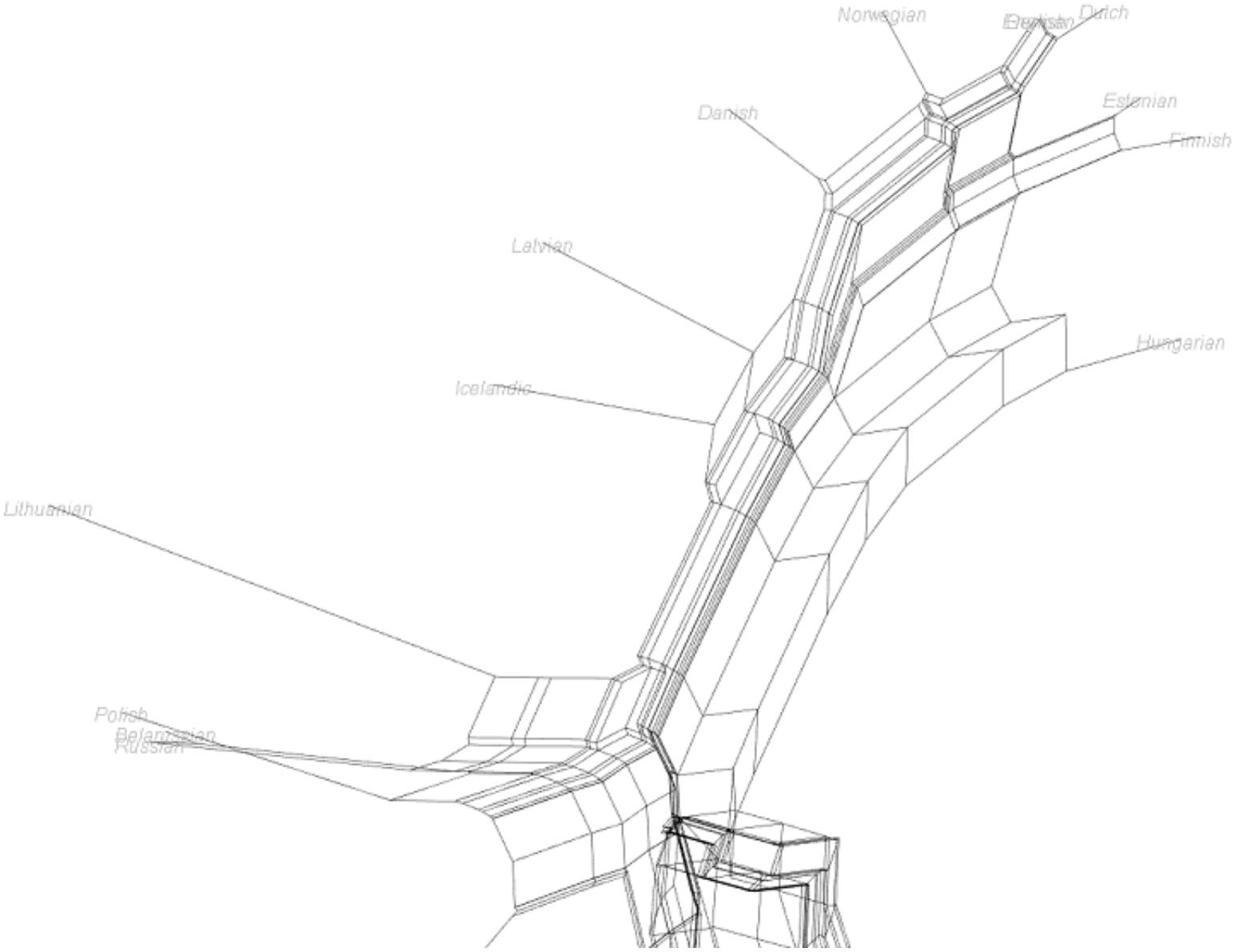
This course concentrates on data exploration, and how different kinds of corpora may support linguistic research. Things you can do with your data to get a better overview, to explore possible relations, to get insight into multi-dimensional structures that the human eye (nor mind) just cannot parse.

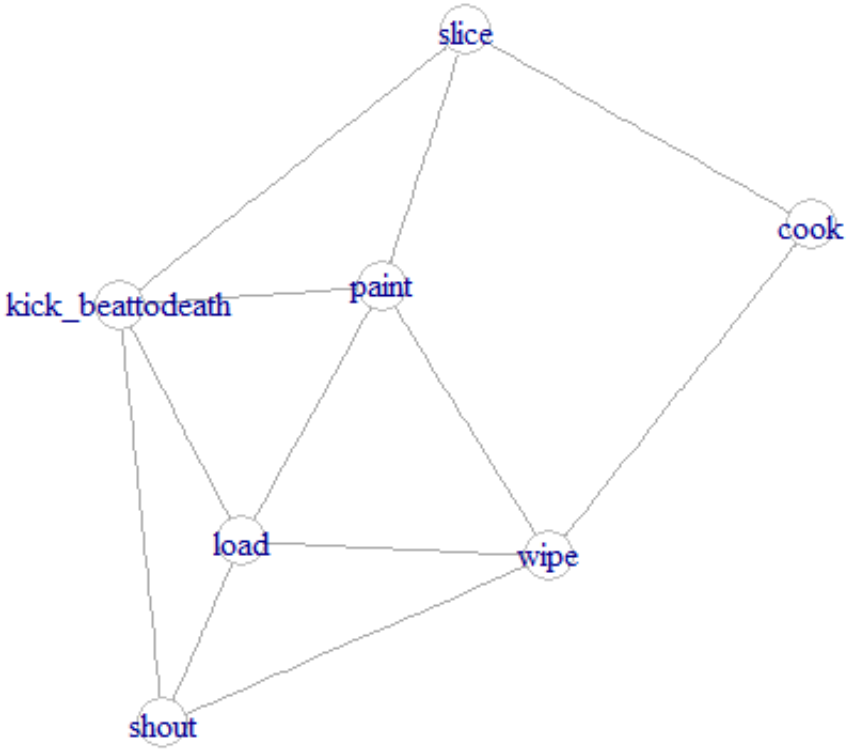
- This course will be entirely hands on (after the first ~15 minutes)
- For that, we'll need tools.
- R and Rstudio, plus additional R packages

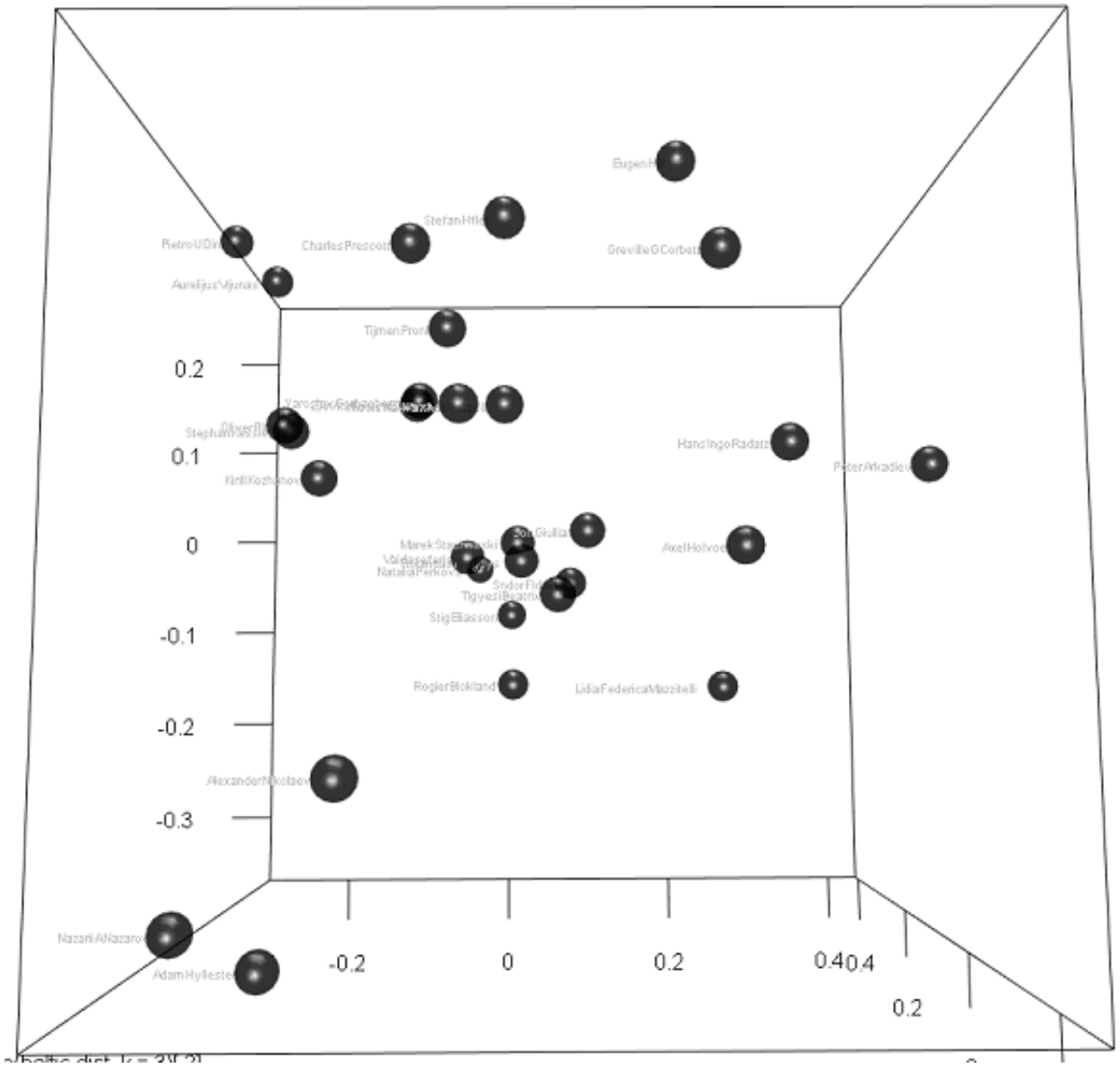
Sneak peek


Height
0
10
20
30
40
50
60
Icelandic
Estonian
Finnish
Danish
Norwegian
Dutch
English
German
Hungarian
Latvian
Macedonian
Bulgarian
Croatian
Irish
Albanian
Bosnian
Turkish
ScottishGaelic
Spanish
Basque
Maltese
French
Greek
Italian
Portuguese
Lithuanian
Belarussian
Russian
Czech
Polish
Cluster Dendrogram

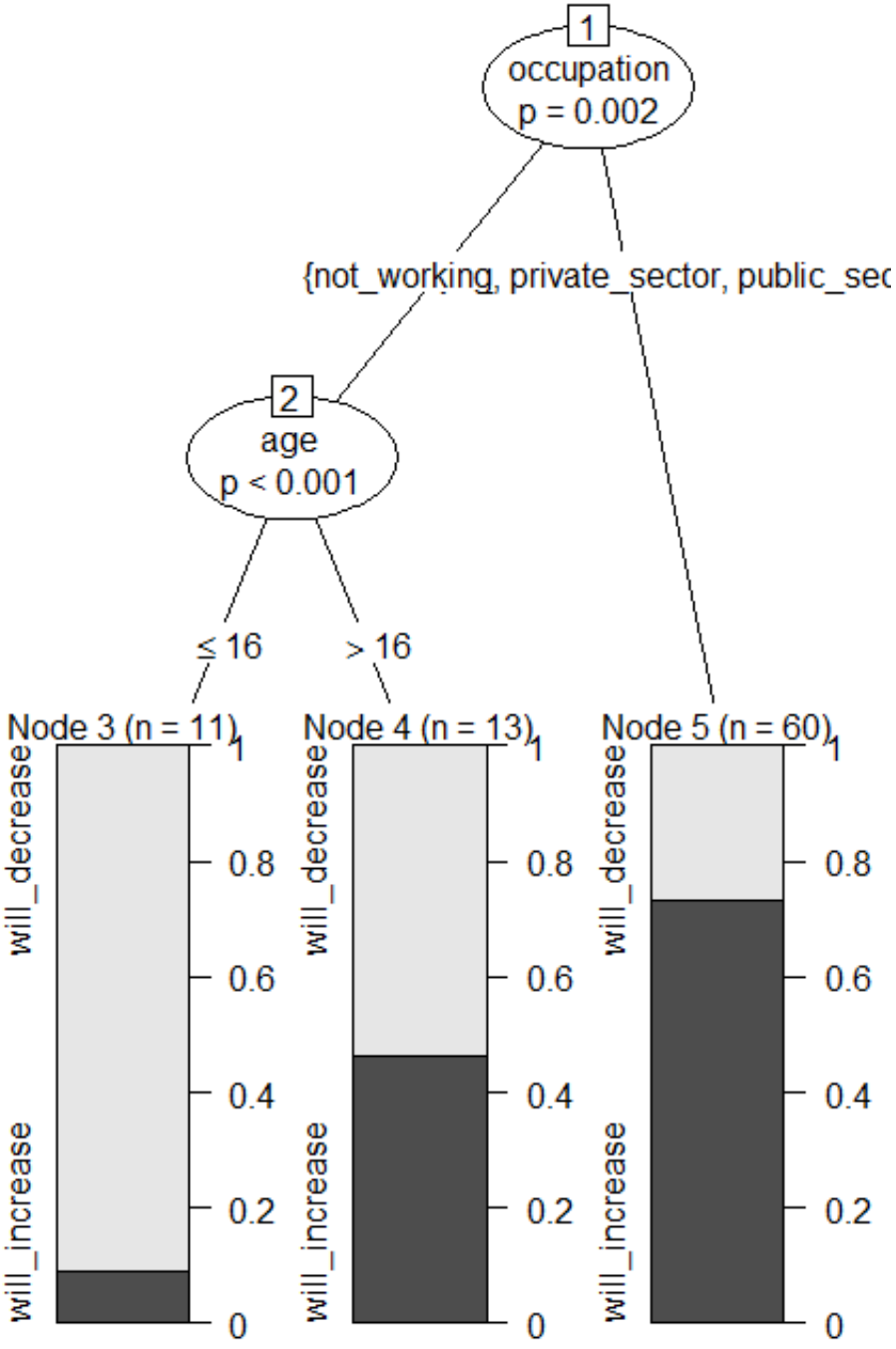

4
run_walk
shoot
hammer_the_metal_flat
5
wipe
rinse
freeze
3
cook
comb
hammer_the_meat_thin
rub
eat_sick
shoot
dance
kick_beatunconscious
1
grind
2
eat_full
pour
load
6
kick_beattodeath
10
5
0
-5
-10
-15
-20
-10
-5
0
5
10
15


Norwegian
Dutch
Danish
Estonian
Finnish
Latvian
Hungarian
Icelandic
Lithuanian
Polish
Russian


slice
cook
kick_beattodeath
paint
load
wipe
shout

- 
0.2
0.1
0
-0.1
-0.2
-0.3
-0.2
0
0.2
0.4
0.4
0.2


germanischen
germanisch
typological
schmalstieg
phonologies
reworked
urgerm
william
areallingvistiska
sound
aspekter
amsterdam
russianisms
germanic
kalbojevalency
linguistic
sprakens
xaldiz
northern
dahl
text
sanskritresearch
sekund
teuer
lithuanian
mariagreek
circum
honor
canonical
indo
bisher
languages
hoch
edited
etymologie
latin
lieber
stephen
philip
grammatical
aume
eher
kallis
autor
other
pietro
osten
baltic
tamm
review
einer
jahre
ymphe
baltisch
reference
linguistics
relations
perspectives
realization
common
koptjevskaja
multiple
fund
european
possibly
baltischen
ruki
forms
spirit
lieb
ambha
argument
encoding
mossman
origin
baldi
unbekannter
root
water
with
semantika
language
philadelphia
dini
uberlieferung
dictionaries
sprache
areal
typologiska
their
fonologi
section
overlooked
nordeuropeiska
slavic
adjektive
ostseefinnisch


1
occupation
p = 0.002
{not_working, private_sector, public_sec
2
age
p < 0.001
≤ 16
> 16
Node 3 (n = 11)
Node 4 (n = 13)
Node 5 (n = 60)
will_decrease
will_increase
1
0.8
0.6
0.4
0.2
0

-

```
grep("`[tT]he storm of[ a-z]* [0-9]{2,4} was (a|the)* [a-z]* .*
```

```
salosGuessLanguage(
    "Oh I wonder what language could this sentence be written in?
    Is it perhaps Old Norse? It sure sounds like a Germanic language,
    but doesn't have any of the funky umlauts.",
     trainingset = ngram.matrix, correct = "eng2011")

               Similarity Correct
eng2011        0.31554233       +
deu2000        0.17583836
deu1871        0.17474825
spa1995        0.12618061
tokpisin       0.10059775
lit1999        0.08051727
ekk1997        0.07286052
pol2013        0.05488152
lav1997        0.05129699
churchslavonic 0.00000000
rus2011        0.00000000
```

Why R?

Why R? For linguistics, and science in general?

- R is a fully functional programming language and programming environment
- Strong support for statistics, data science and making pretty graphs
- Highly extensible: an ever-growing number of user-created packages
  - *less work for the user; also a smoother learning curve*
  - *"a package for everything" not far from the truth*
  - *e.g., these slides are made in R using the relevant extensions*
- Large user base (meaning lots of useful info on the web)
  - *not just academics: companies like Microsoft and Google use R*
- Free and open source (unlike Matlab, Stata, SPSS, Excel, etc.)

An example of replicability

- Everything you do is consistently replicable (both by you and others, if they have the same code and data)
  - *Code written years ago (usually) runs today, and will run tomorrow as well*
  - *not bound by GUIs that tend to change every half a decade*

An example of replicability

Jan Vanhove and Raphael Berthele 2014. The lifespan development of cognate guessing skills in an unknown related language. International Review of Applied Linguistics in Language Teaching (IRAL), 53(1), 1-38.

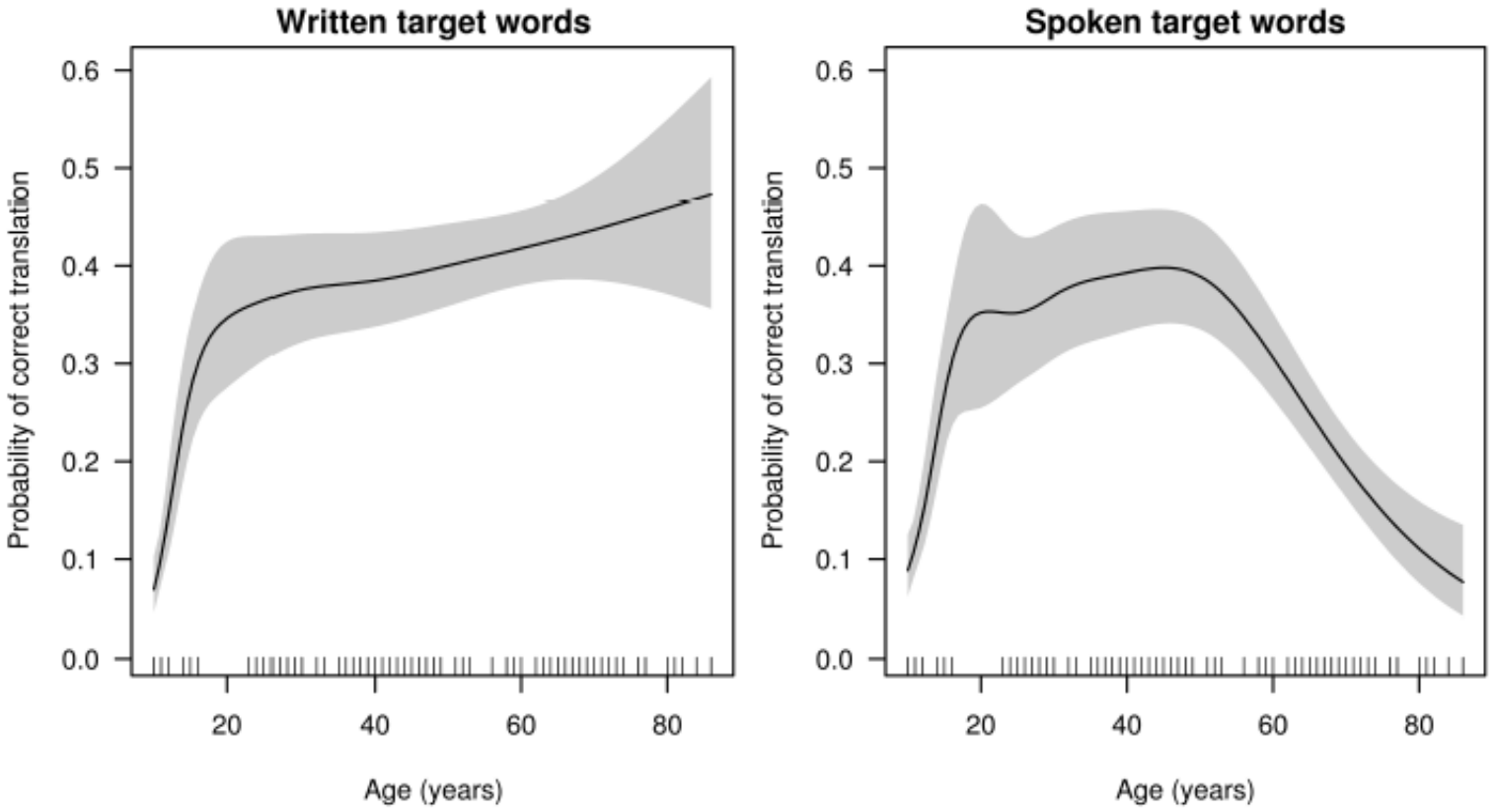


Fig. 2: Age trends for GAMM-fitted translation accuracy in the written (left) and spoken modality (right) with 95% confidence bands. The trend lines represent the modelled probabilities for a typical female participant who did not translate any profile word correctly.

The data

http://figshare.com/articles/Data_and_code_for_The_lifespan_development_of_cognate_guessing_skills_in_an_unknown_related_language/936924

	Stimulus	Subject	Block	Modality	Translation	Correct	Sex	Age	NrLang	BWDS	WST
1	agg	1034	1	Auditory	eng	0	female	51	3	6	3
2	agg	2151	2	Auditory		0	female	26	4	11	3
3	agg	9022	2	Auditory	ende	0	female	65	2	7	3
4	agg	7337	2	Auditory	Tinte	0	female	49	3	5	3
5	agg	8477	1	Auditory		0	female	42	3	8	3
6	agg	6544	1	Auditory	eng	0	female	16	4	6	2
7	agg	5153	1	Auditory	eng	0	female	51	3	6	3
8	agg	1668	1	Auditory	Tinte	0	female	37	3	6	3
9	agg	288	2	Auditory	eng	0	male	28	4	9	3
10	agg	9781	2	Auditory	tinte	0	male	43	4	12	3
11	agg	7886	2	Auditory		0	female	10	1	6	2
12	agg	7164	1	Auditory		0	male	10	1	2	
13	agg	5566	2	Auditory	ich	0	female	42	4	4	3
14	agg	447	2	Auditory	Tinte	0	female	53	3	8	3
15	agg	230	1	Auditory		0	male	84	4	7	3
16	agg	134	2	Auditory		0	male	33	3	6	3
17	agg	8674	1	Auditory	eng	0	female	63	3	6	3
18	agg	9318	1	Auditory	ich	0	male	72	2	6	3

The code

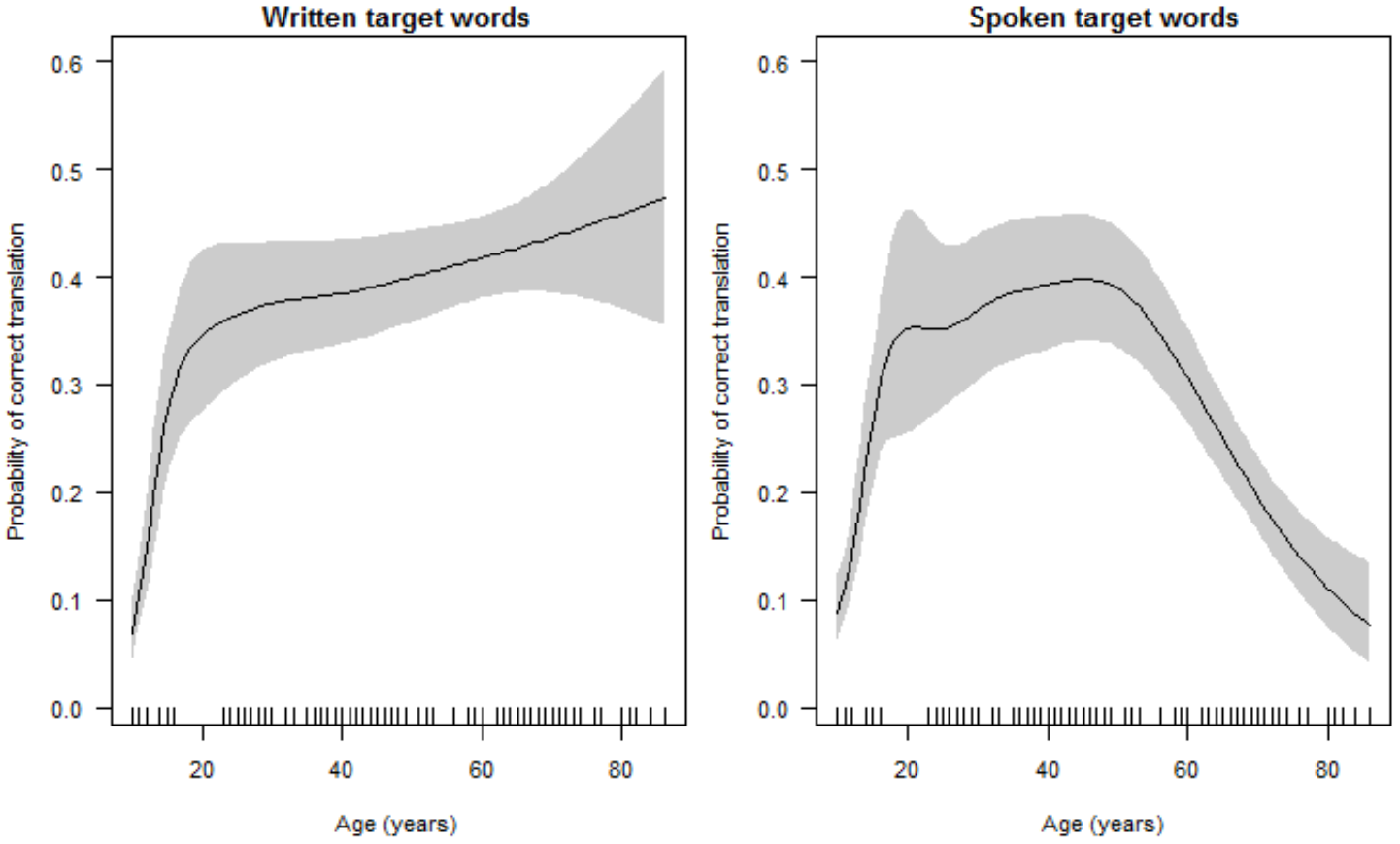
```
228 # GAMM written
229 age_written = bam(Correct ~ s(Age, bs="ad") + Sex + ProfileRight + s(Subject, bs="re") + s(Stimulus, bs
    ="re"), data=dat_written, family=binomial)
230 age_written2 = bam(Correct ~ s(Age, bs="ad") + ProfileRight + s(Subject, bs="re") + s(Stimulus, bs="re"
    ), data=dat_written, family=binomial)
231
232 # GAMM spoken
233 age_spoken = bam(Correct ~ s(Age, bs="ad") + Sex + ProfileRight + s(Subject, bs="re") + s(Stimulus, bs
    ="re"), data=dat_spoken, family=binomial)
234 age_spoken2 = bam(Correct ~ s(Age, bs="ad") + s(Subject, bs="re") + s(Stimulus, bs="re"), data=dat_spoke
    n, family=binomial)
235
236 rm(age_written2, age_spoken2)
237 # Plot
238 #png("gamms.png", width=5, height=7, units="in", res=1200)
239 #par(mfrow=c(2,1), mar=c(4,4,2,0.5)+0.1, cex=0.7)
240 png("gamms.png", width=6.5, height=3.5, units="in", res=1200)
241 par(mfrow=c(1,2), mar=c(4,4,1.5,0.5)+0.1, las=1, cex=0.7)
242 plot.gam(age_written, select=1, shade=T, trans=plogis, shift=-0.7375,
243          xlab="Age (years)",
244          ylab="Probability of correct translation", ylim=c(-5,0.4054651)+0.7375,
245          main="Written target words",
246          las=1)
247 plot.gam(age_spoken, select=1, shade=T, trans=plogis, shift=-1.04199,
248          xlab="Age (years)",
249          ylab="Probability of correct translation", ylim=c(-5,0.4054651)+1.04199,
250          main="Spoken target words",
251          las=1)
252 par(mfrow=c(1,1))
253 dev.off()
```

233:144 (Untitled) R Script

The analysis reproduced in R

```
> summary(age_spoken)

Family: binomial
Link function: logit

Formula:
Correct ~ s(Age, bs = "ad") + Sex + ProfileRight + s(Subject,
    bs = "re") + s(Stimulus, bs = "re")

Parametric coefficients:
                Estimate Std. Error z value Pr(>|z|)
(Intercept)     -1.04609    0.37568  -2.785  0.00536 **
Sexmale         -0.06311    0.13682  -0.461  0.64462
ProfileRightTRUE  0.28808    0.17896   1.610  0.10746
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Approximate significance of smooth terms:
               edf  Ref.df  Chi.sq p-value
s(Age)       4.967   5.306   85.95  <2e-16 ***
s(Subject)  104.722 155.000  454.83  <2e-16 ***
s(Stimulus)  40.139  41.000 1952.10  <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

R-sq.(adj) =   0.51   Deviance explained = 46.5%
fREML =   9404  Scale est. = 1         n = 6678
```

The plots recreated in R



The R language is not that different from a natural language

- Vocabulary: predominantly borrowed from English
- Word order: fixed (but free "word order" among the list of named parameters)
- Embedded clauses are pretty common, e.g., `plot( hclust( dist( data) ) ) )`
- From a natural language perspective, the communication consist of commands and/or questions.
- For example: "what is the sum of 1 and 4?"
  - *`sum(1, 4)`*
- "Make an object and call it 'mynumber'. Have the object contain the number 7"
  - *`mynumber = 7`*

Let’s try another one

Let's try another one

I have a *numeric* vector of observations of the weights of cats

```
cat.weights = numeric()
```

…which consists of the values 1, 3, 2.5, 4, 1, 2

```
cat.weights = c(1, 3, 2.5, 4, 1, 2)
```

…and another with their heights, 12, 13, 14, 15, 11, 12

```
 cat.heights = c(12, 13, 14, 15, 11, 12)
```

Do tell me, what is the *correlation* between those two vectors?

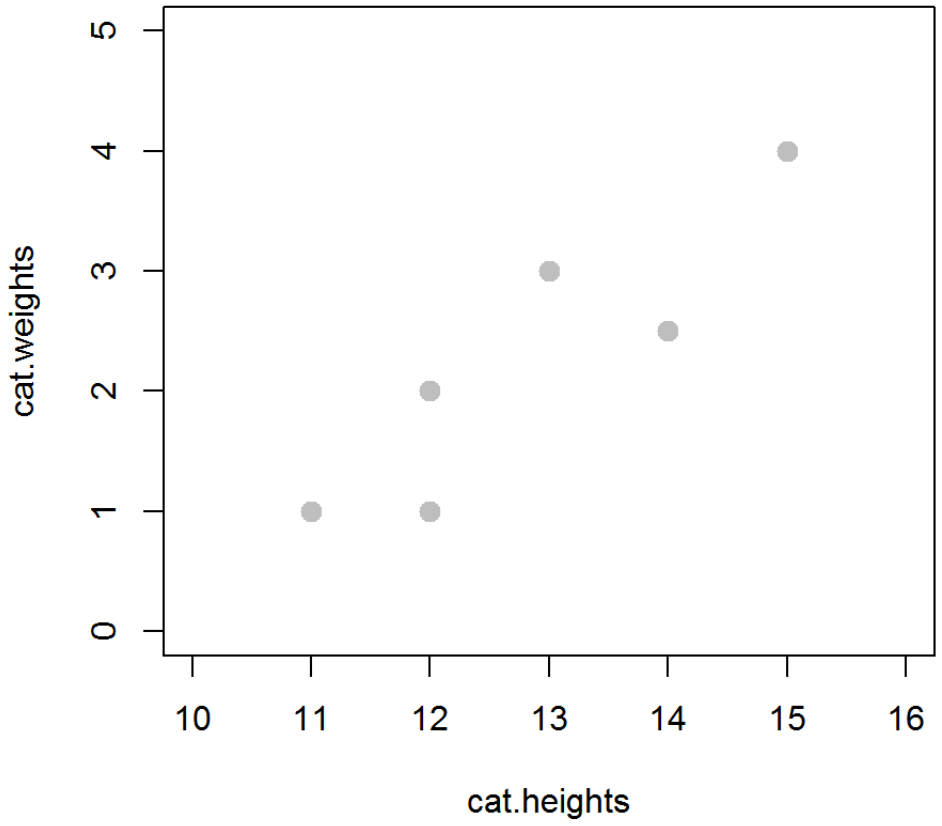
```
cor(cat.weights, cat.heights)
```

```
## [1] 0.8980165
```

• • •

Would you be so kind dear and *plot* me these observations…

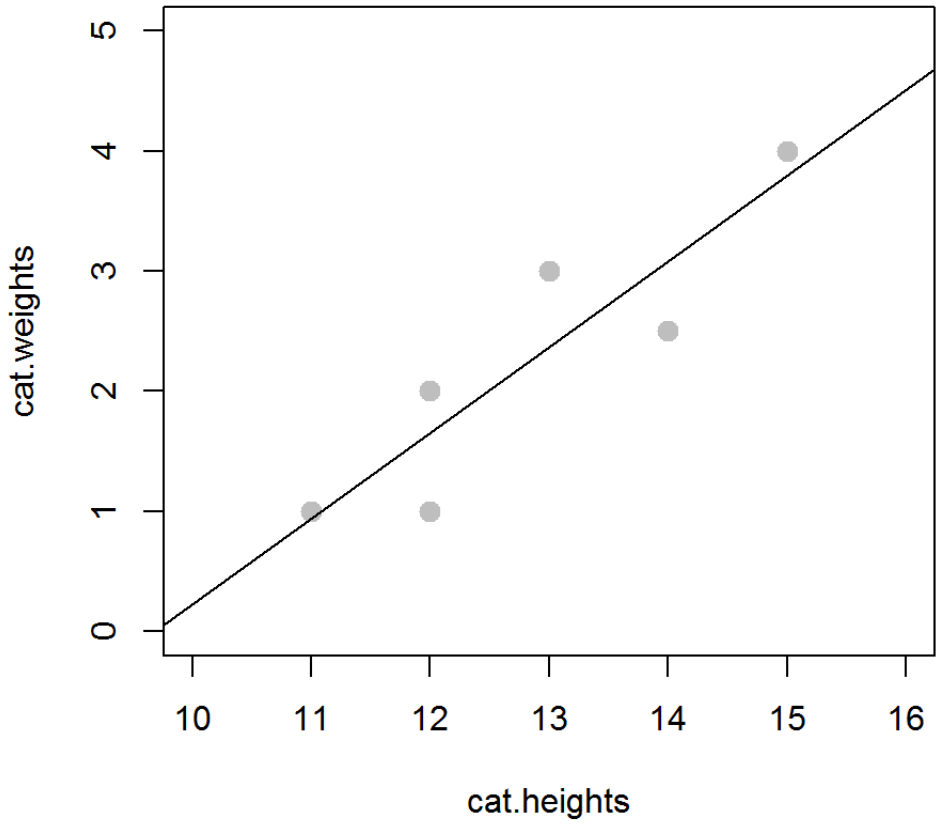
```
plot(cat.weights ~ cat.heights)
```



• • •

and draw a *linear regression line* on the plot to illustrate the correlation? Thanks!

```
abline(lm(cat.weights ~ cat.heights))
```



• • •

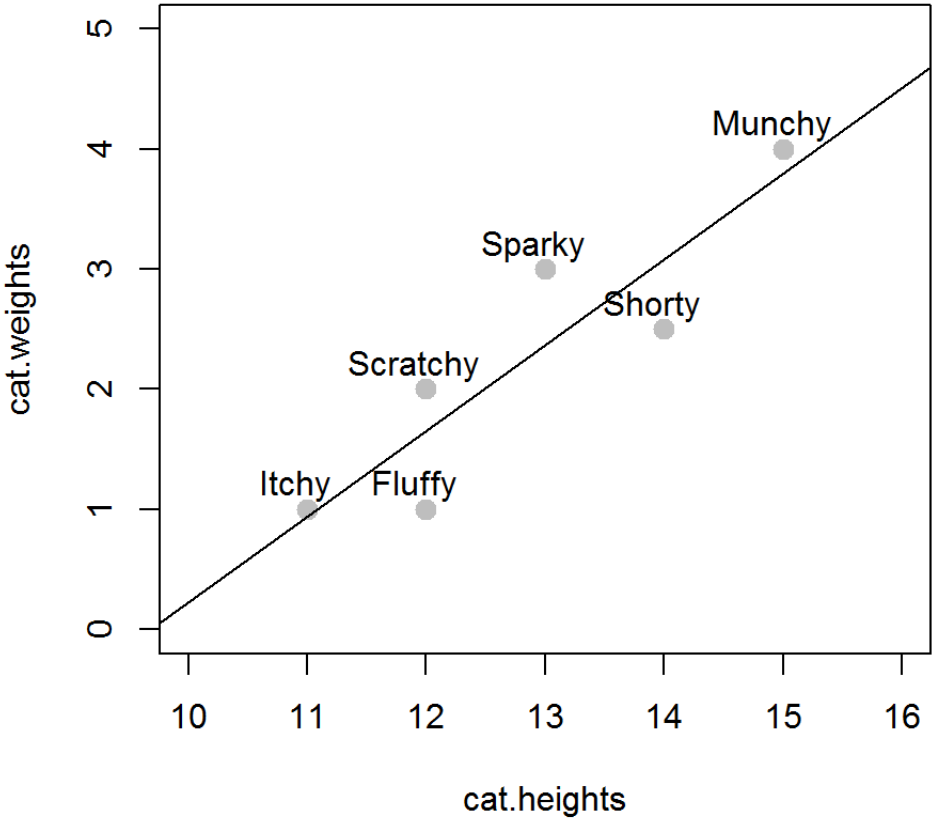
But the six observations can also be characterized by name, so let's call them Fluffy, Sparky, Shorty, Munchy, Itchy and Scratchy.

```
catnames = c("Fluffy", "Sparky", "Shorty", "Munchy",
             "Itchy", "Scratchy")
```

• • •

Add these *labels* to the plot as *text*, at the relevant *x* and *y* coordinates.

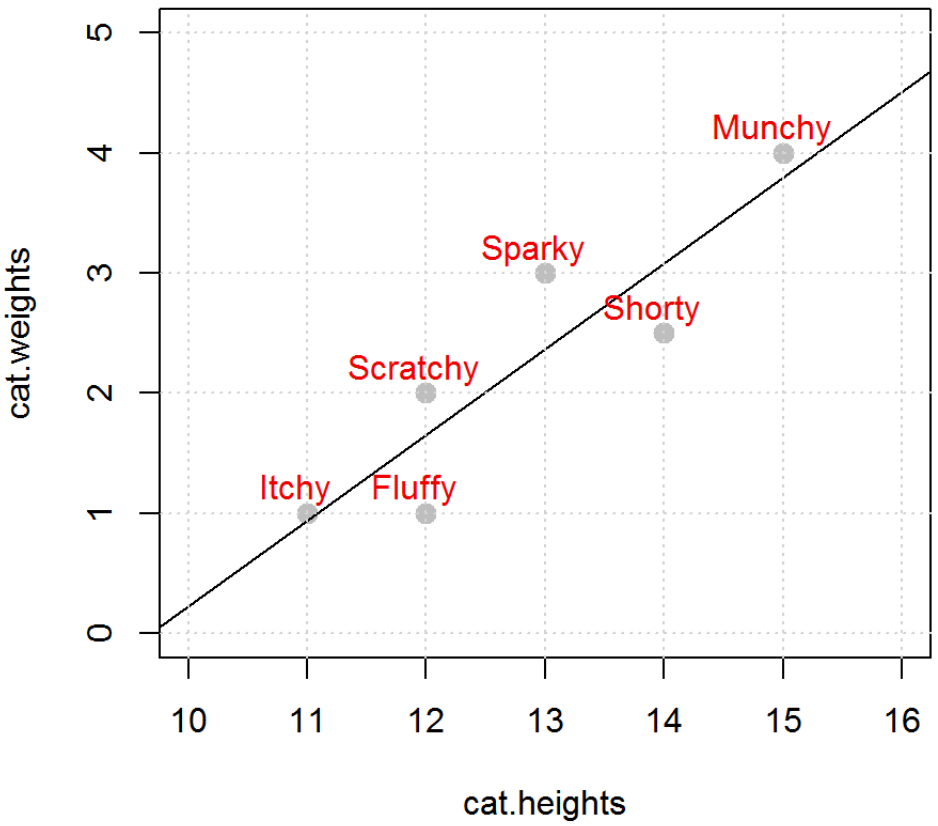
```
text(x = cat.heights, y = cat.weights,
     labels = catnames)
```


Munchy
Sparky
Shorty
Scratchy
Itchy
Fluffy
cat.weights
cat.heights
0
1
2
3
4
5
10
11
12
13
14
15
16

• • •

Hey I also like red, so do *color* them *red*, and add a *grid* too. Sweet!

```
text(x = cat.heights, y = cat.weights,
     labels = catnames, col = "red"); grid()
```


cat.weights
cat.heights
0
1
2
3
4
5
10
11
12
13
14
15
16
Munchy
Sparky
Shorty
Scratchy
Itchy
Fluffy

• • •

Oh and while your're at it, please *print* all the possible combinations of heights and weights in this sample.

```
for(w in cat.weights) {
    for(h in cat.heights)
        print(c(w, " & ", h), quote = FALSE)
}
```

```
## [1] 1    &  12
## [1] 1    &  13
## [1] 1    &  14
## [1] 1    &  15
## [1] 1    &  11
## [1] 1    &  12
## [1] 3    &  12
## [1] 3    &  13
## [1] 3    &  14
## [1] 3    &  15
## [1] 3    &  11
## [1] 3    &  12
## [1] 2.5  &  12
## [1] 2.5  &  13
## [1] 2.5  &  14
## [1] 2.5  &  15
## [1] 2.5  &  11
## [1] 2.5  &  12
## [1] 4    &  12
## [1] 4    &  13
## [1] 4    &  14
## [1] 4    &  15
## [1] 4    &  11
## [1] 4    &  12
```

```
## [1] 1    &  12
## [1] 1    &  13
## [1] 1    &  14
## [1] 1    &  15
## [1] 1    &  11
## [1] 1    &  12
## [1] 2    &  12
## [1] 2    &  13
## [1] 2    &  14
## [1] 2    &  15
## [1] 2    &  11
## [1] 2    &  12
```

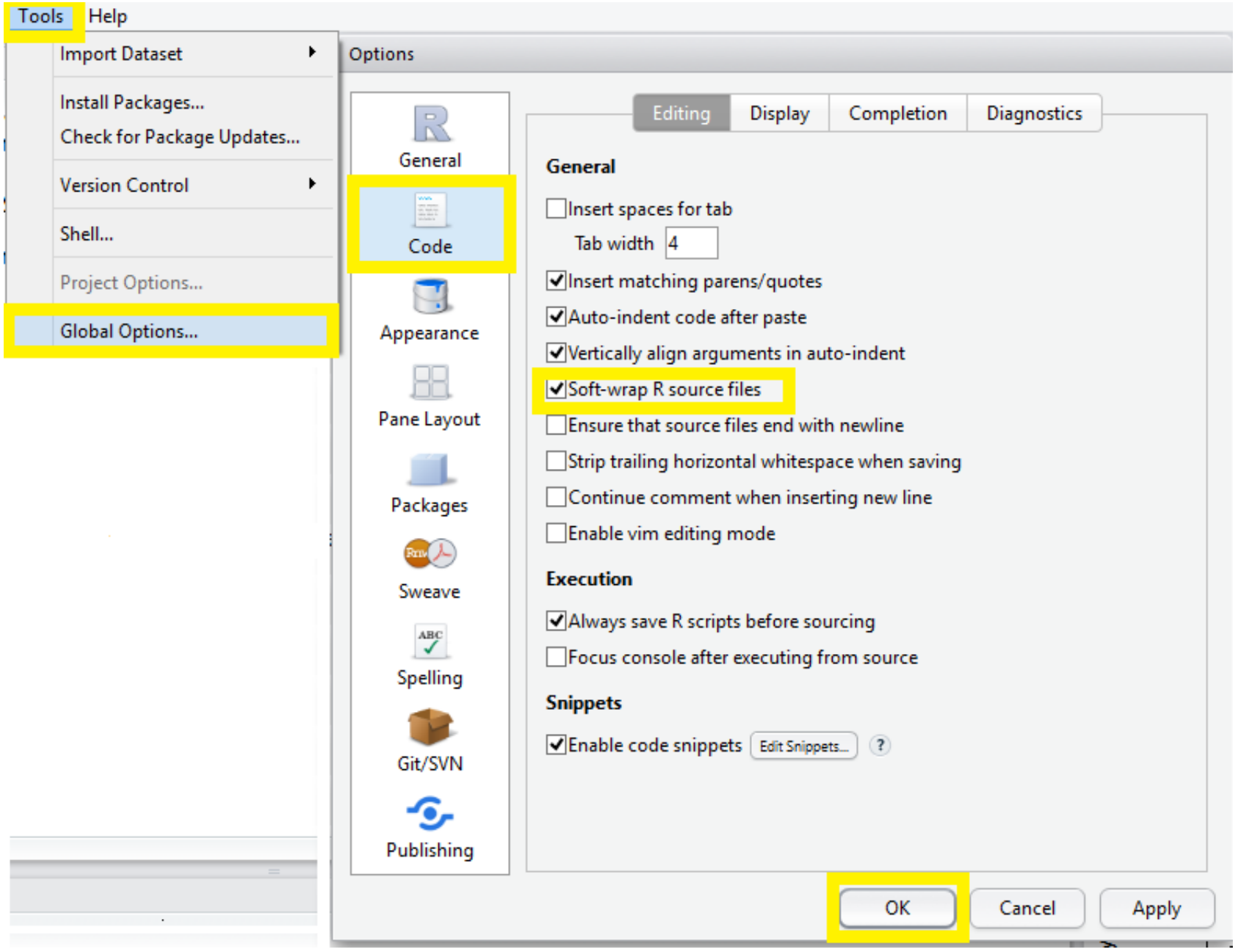
How this course is structured

Two tracks, A and B. Same room, same data, broadly the same tools, but difference in difficulty.

- Track A: no programming or stats knowledge assumed.
- Track B: includes all the tasks of Track A, plus more advanced tasks, some of which require programming your own functions in R.
- If you are done with Track A tasks in a given section and you have time, feel free to try out the tasks marked 'Track B'
- The course is modular. If it seems worthwhile to spend more time with any given task, then we can skip another one.
- It will be possible to organize an extra informal session on Friday, if anyone is interested. The extra time could be used to cover any tasks that were skipped due to time constraints (if any), or as a consultation, in case you have further questions regarding the methods discussed in the course, or about what could be applied to your own data, etc.

Let’s get started; open RStudio.

Tools (top menu bar) > Global Options > Code >> Soft-wrap R source files


Tools
Help
Import Dataset
Install Packages...
Check for Package Updates...
Version Control
Shell...
Project Options...
Global Options...
Options
General
Code
Appearance
Pane Layout
Packages
Sweave
Spelling
Git/SVN
Publishing
Editing
Display
Completion
Diagnostics
General
Insert spaces for tab
Tab width 4
Insert matching parens/quotes
Auto-indent code after paste
Vertically align arguments in auto-indent
Soft-wrap R source files
Ensure that source files end with newline
Strip trailing horizontal whitespace when saving
Continue comment when inserting new line
Enable vim editing mode
Execution
Always save R scripts before sourcing
Focus console after executing from source
Snippets
Enable code snippets
Edit Snippets...
OK
Cancel
Apply