

Session 3

Lunch R workshop 2018.03.20

Max Lindmark & Philip Jacobson

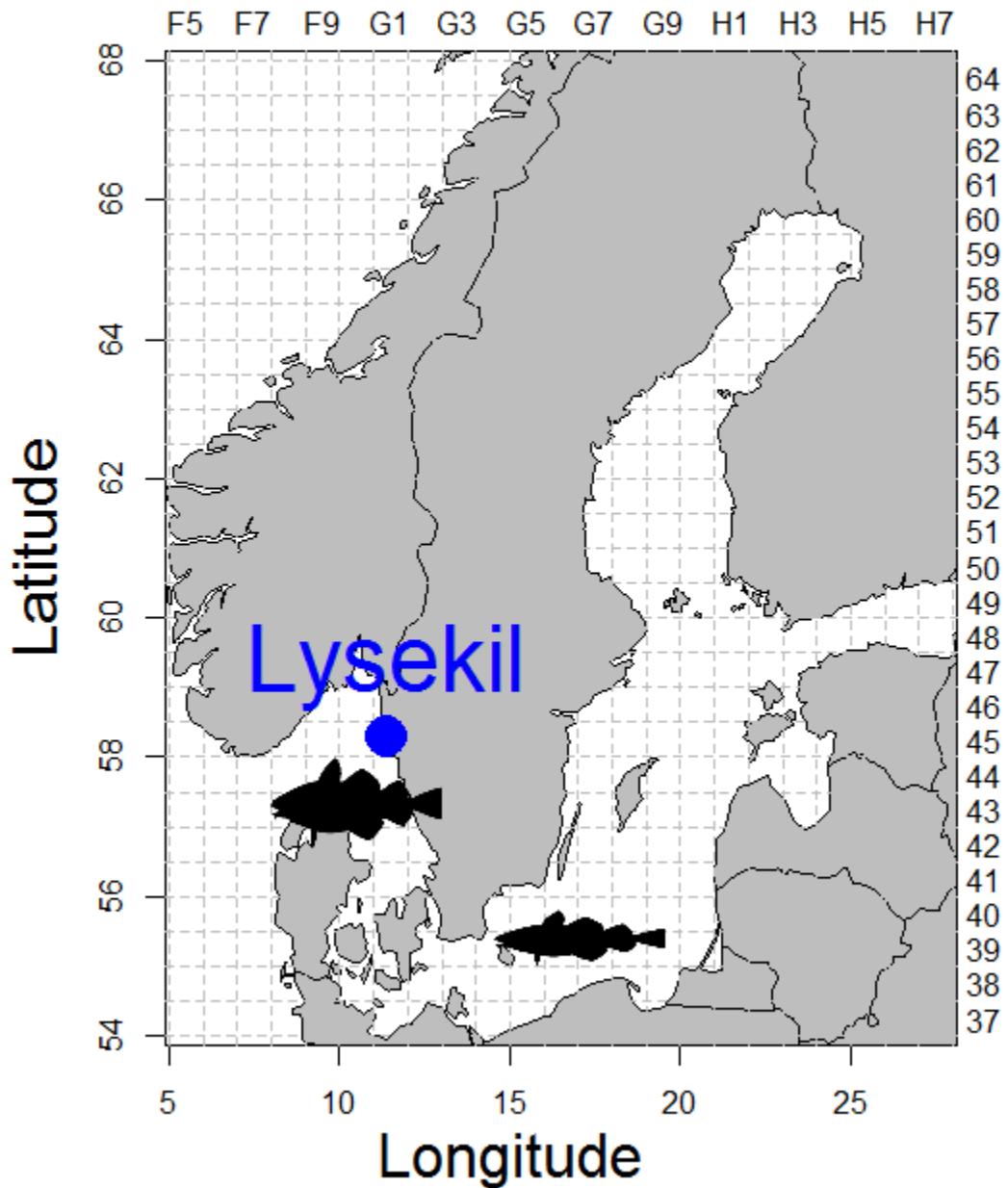
lunchR



<https://www.slideshare.net/continuumio/open-data-science-with-r-and-anaconda>

Sessions

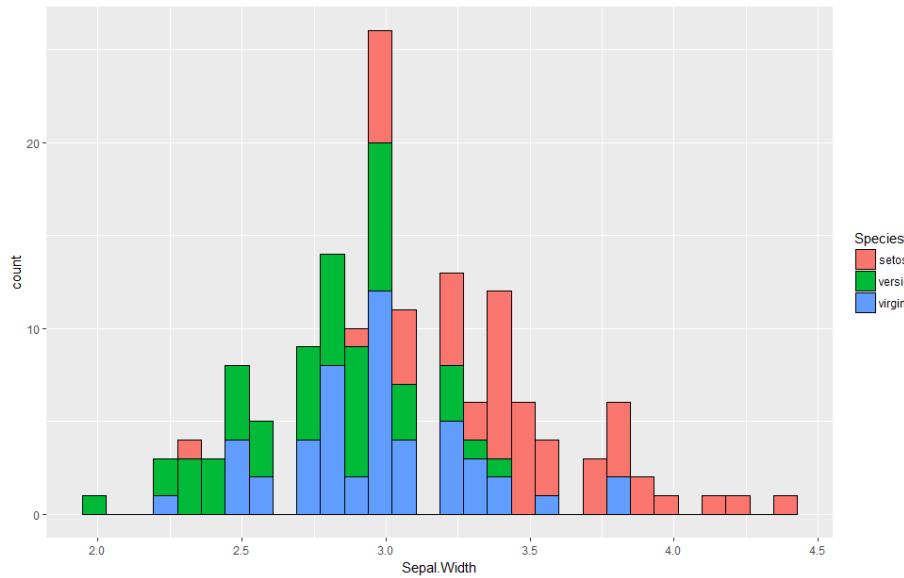
1. Intro to why R, data formats and plotting styles in R
2. Digging deeper into data exploration in R using base R and ggplot; focus on different graph styles
- 3. Tips and tricks in R – format your data for easy plotting**
4. Creating publish-quality figures



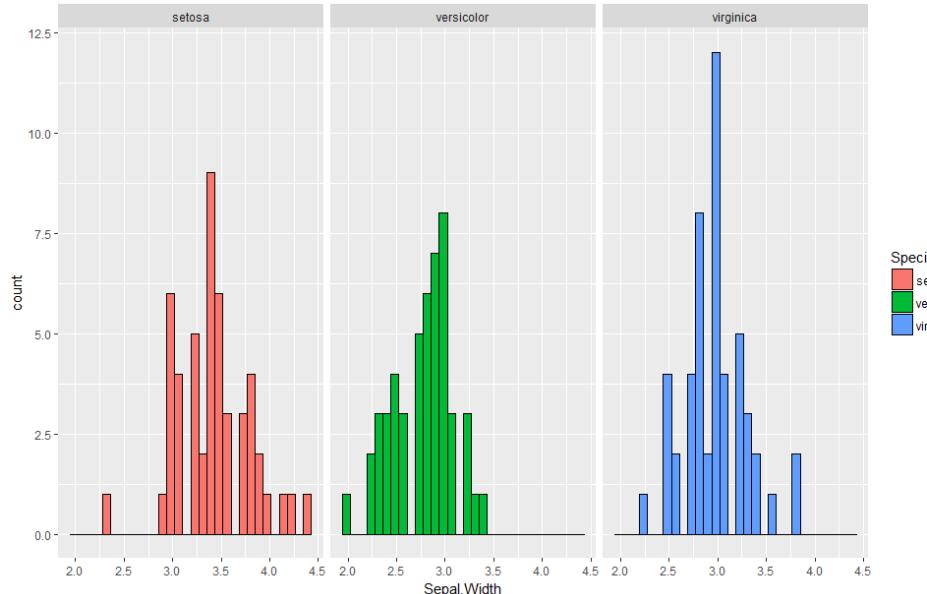
Session 2: Some of you had problems creating the final map

- Windows 10 users.
- Apparently some package in the tidyverse “masked” functions in the map and/or mapplots packages.
- Restart R (or the computer) and then only use `library(map)` and `library(mapplots)` in the beginning and see if the code works.

We produced this plot



Same thing but in panels

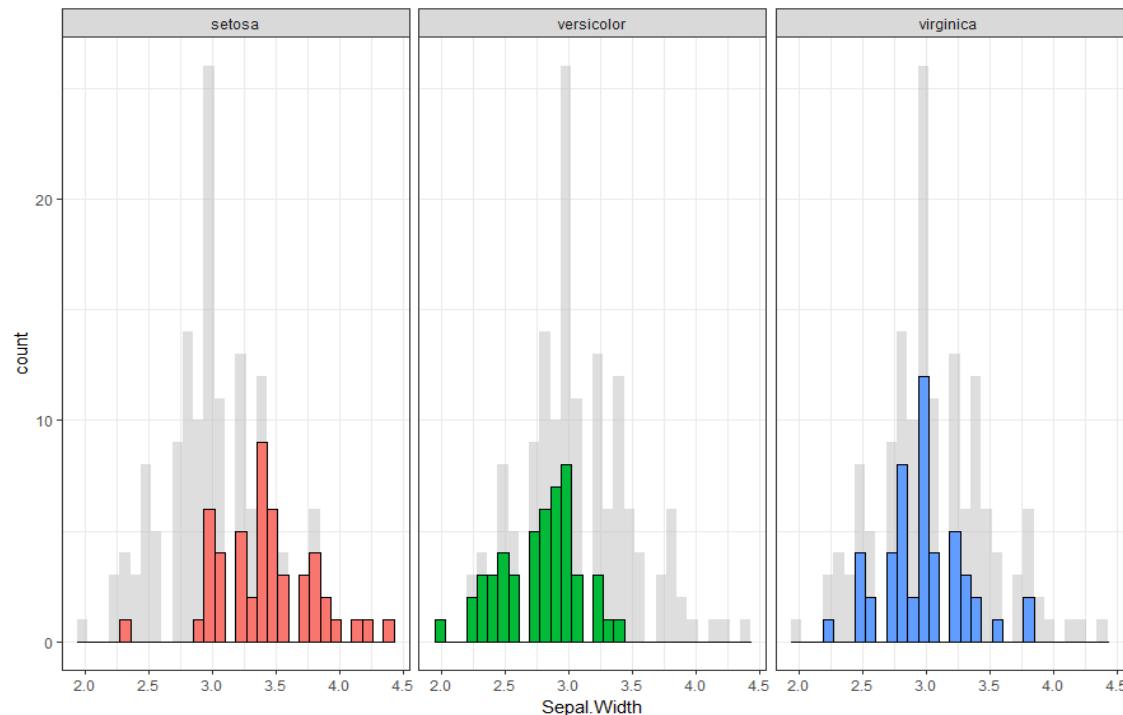


lunchR

Session 2: One more thing..

<https://drsimonj.svbtle.com/plotting-background-data-for-groups-with-ggplot2>

Same thing in panels with the background data in grey (i.e. the first plot)



Sessions

1. Intro to why R, data formats and plotting styles in R
2. Digging deeper into data exploration in R using base R and ggplot; focus on different graph styles
- 3. Tips and tricks in R – format your data for easy plotting**
4. Creating publish-quality figures

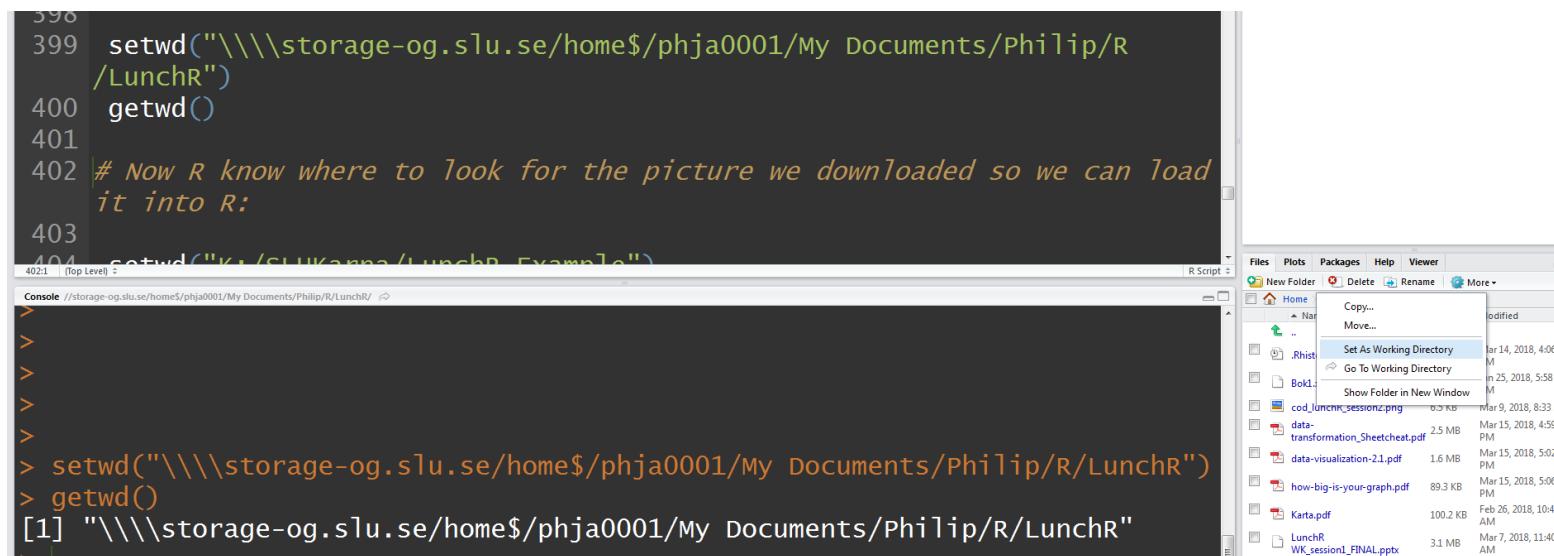
Tips and tricks in R – format your data for easy plotting

- Get your data into R and share it with others
 - Email, Server or URL?
- Handle, re-organize and create data in R
 - Pipes and ggplot = goosebumps
- Cheatsheets

How and why do we share data and scripts?

- Co-authors
- Back up
- Reviewers
- Students: teaching, R-labs, R-courses (e.g. LunchR)

How do we share data and scripts? – Session 2



The screenshot shows the RStudio interface. On the left, the R Script pane displays the following R code:

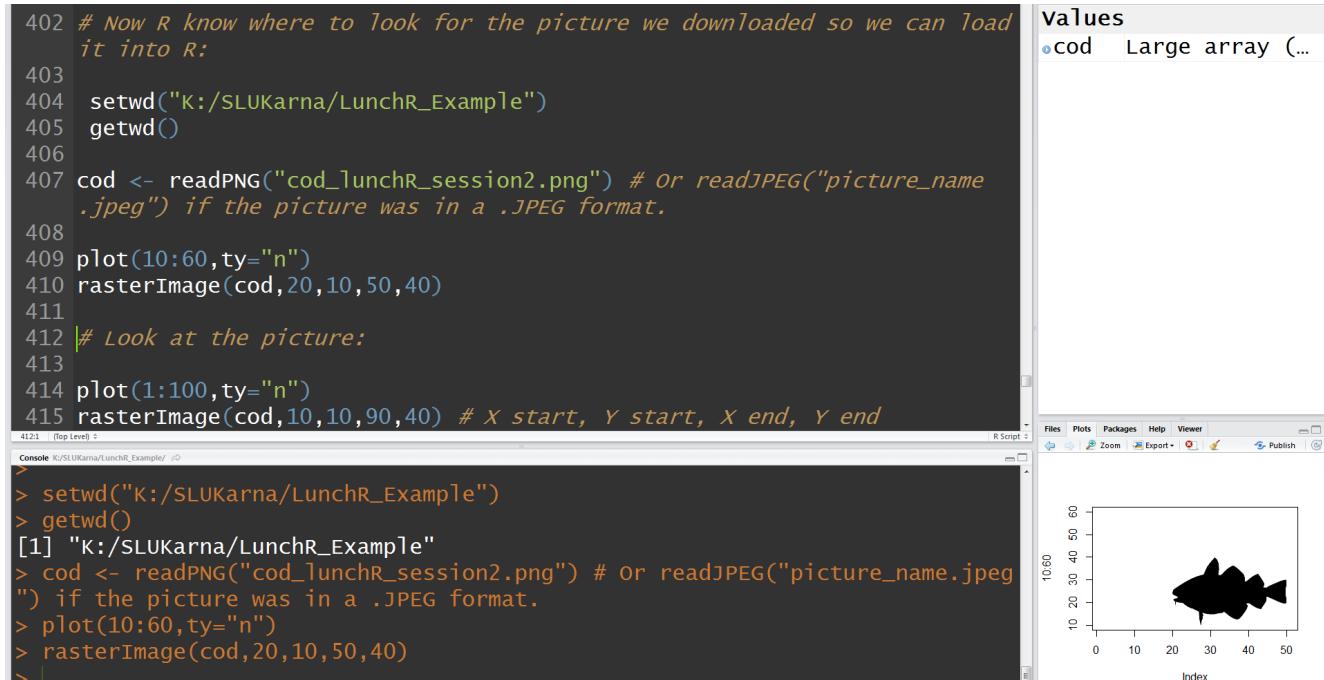
```
398 setwd("\\\\storage-og.slu.se/home$/phja0001/My Documents/Philip/R  
/LunchR")  
400 getwd()  
401  
402 # Now R know where to look for the picture we downloaded so we can load  
it into R:  
403  
404 setwd("K:/SLU/Kappa/LunchR_Example")
```

On the right, the File Explorer pane shows a directory structure with files like .Rhist, Box1, cod_lunchr_session2.png, data-transformation_Sheetcheat.pdf, data-visualization-2.1.pdf, how-big-is-your-graph.pdf, Karta.pdf, and LunchR_WK_session1_FINAL.pptx.

This is not bad but it can lead to problems. You need to email scripts+data and the user need to manually change the code.

For anyone else to use this script, the person needs to specify where they have stored the object (data or picture, Session 2) we need to read in to R.

How do we share data and scripts?



The screenshot shows the RStudio interface. On the left, the R Script pane contains the following R code:

```
402 # Now R know where to look for the picture we downloaded so we can load  
# it into R:  
403  
404 setwd("K:/SLUKarna/LunchR_Example")  
405 getwd()  
406  
407 cod <- readPNG("cod_lunchR_session2.png") # or readJPEG("picture_name.  
.jpeg") if the picture was in a .JPEG format.  
408  
409 plot(10:60,ty="n")  
410 rasterImage(cod,20,10,50,40)  
411  
# Look at the picture:  
412  
413  
414 plot(1:100,ty="n")  
415 rasterImage(cod,10,10,90,40) # X start, Y start, X end, Y end
```

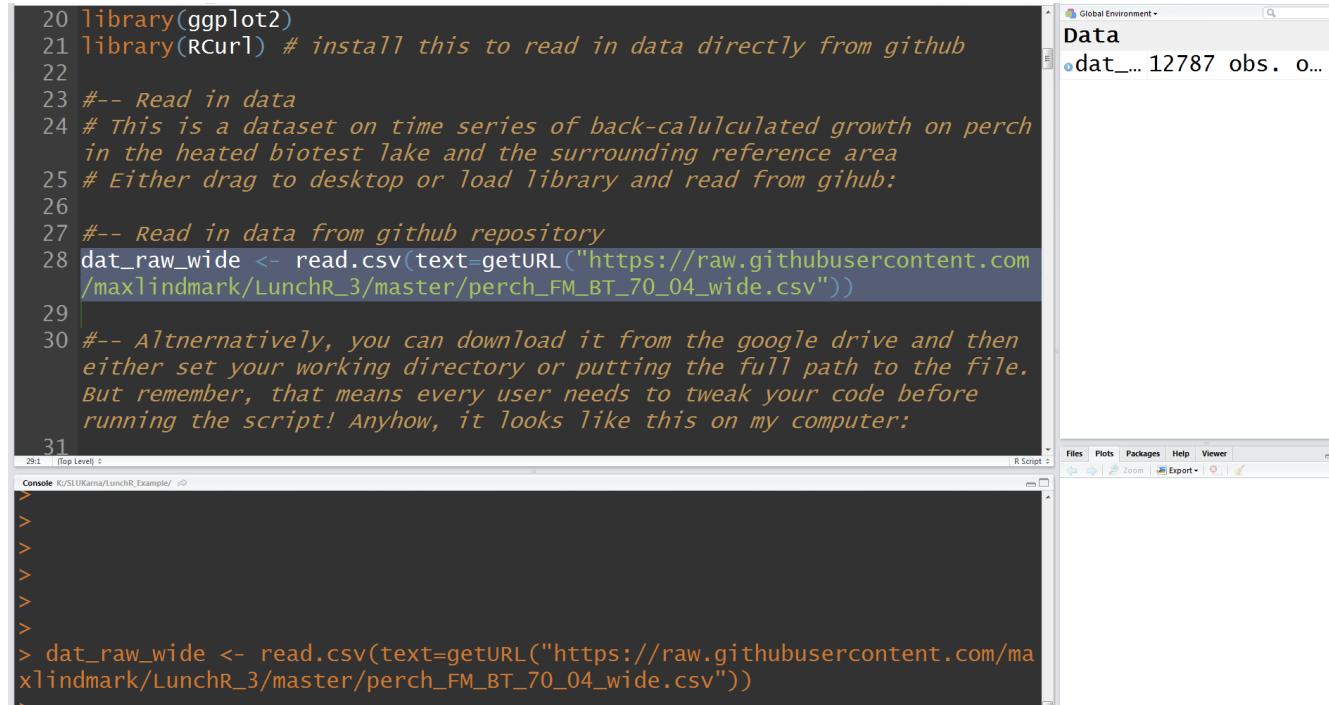
On the right, the RStudio interface shows the following:

- A "values" sidebar with a single entry: "cod Large array (...)".
- An "Index" plot window showing a grayscale image of a fish against a white background. The x-axis is labeled "Index" and ranges from 0 to 50. The y-axis is labeled "Index" and ranges from 10 to 60.

For those that do not have access to the server?
– To bad..

We can use a server. Now everyone that have access to the Öregrund common server K: can produce this plot by using the script.

How do we share data and scripts?



The image shows a screenshot of the RStudio interface. On the left, the 'R Script' pane displays R code for reading data from GitHub. The code includes comments explaining the process. On the right, the 'Global Environment' pane shows a data frame named 'dat' with 12787 observations.

```
20 library(ggplot2)
21 library(RCurl) # install this to read in data directly from github
22
23 #-- Read in data
24 # This is a dataset on time series of back-calculated growth on perch
# in the heated biotest lake and the surrounding reference area
25 # Either drag to desktop or load library and read from github:
26
27 #-- Read in data from github repository
28 dat_raw_wide <- read.csv(text=getURL("https://raw.githubusercontent.com/
maxlindmark/LunchR_3/master/perch_FM_BT_70_04_wide.csv"))
29
30 #-- Alternatively, you can download it from the google drive and then
# either set your working directory or putting the full path to the file.
# But remember, that means every user needs to tweak your code before
# running the script! Anyhow, it looks like this on my computer:
31
```

Console output shows the command being run:

```
>
>
>
>
>
> > dat_raw_wide <- read.csv(text=getURL("https://raw.githubusercontent.com/
maxlindmark/LunchR_3/master/perch_FM_BT_70_04_wide.csv"))
<
```

URL! We can upload our script and data to an URL like github.org and load the data directly from there (we will do this later today).

Available to everyone with an internet connection

How do we share data and scripts?

This screenshot shows Max Lindmark's GitHub profile. At the top, there is a large portrait photo of him. Below it, his name "Max Lindmark" and GitHub handle "maxlindmark" are displayed. A brief bio states he is a Ph.D. student in marine ecology at the Swedish University of Agricultural Sciences. Under "Popular repositories", three repos are listed: "maxlindmark" (empty), "perch_growth_lat" (script for inspecting summarized length at age data on perch along a latitudinal gradient in the Baltic Sea, written in R), and "LunchR_3". A heatmap below shows "91 contributions in the last year" from March 2018, with most activity in January and February. At the bottom, it says "Created 9 commits in 1 repository" for "maxlindmark/LunchR_3".

This screenshot shows the "LunchR_3" repository page. The top navigation bar includes "Features", "Business", "Explore", "Marketplace", "Pricing", "This repository", "Search", "Sign in or Sign up". The repository details show 0 stars, 0 forks, and 1 contributor (maxlindmark). It features a "Join GitHub today" banner. Below, a summary shows 10 commits, 1 branch, 0 releases, and 1 contributor. A list of recent commits includes:

- maxlindmark Add files via upload (Latest commit ba01fb9 31 seconds ago)
- README.md Update README.md (5 days ago)
- Session_3_LunchR.txt Add files via upload (31 seconds ago)
- perch_FM_BT_70_04_wide.csv Add files via upload (5 days ago)
- README.md (empty file)

<https://github.com/maxlindmark/>

How do we share data and scripts?

Email: Send script+data.

The user need to manually change the working directory =
can be problematic. Automatic back-up.

Server: As long as everyone have access to it, go ahead!
Automatic back-up. No need for manual editing.

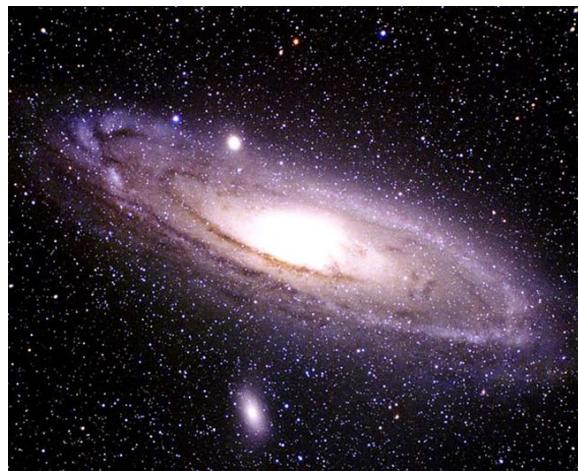
URL: Everyone can access script and data. Automatic
back-up. No need for manual editing.

We will use URL today

Handle, re-organize and create data in R

- Used to be tricky and time consuming
- You had to learn a lot of functions and code
- Very different from other programs (e.g. Excel)

This is not the case anymore thanks to:



Tidy data with one line of code!

“tidy” data

	x	y	group
1	1	0.497868052	a
2	2	0.691927672	a
3	3	0.760313282	a
4	4	0.155401223	a
5	5	0.849457093	a
6	6	0.946817819	a
7	7	0.588419190	a
8	8	0.502250815	a
9	9	0.189779918	a
10	10	0.001836858	a
11	1	1.077578062	b
12	2	0.334111338	b
13	3	0.222741224	b
14	4	1.139136706	b
15	5	0.492948723	b
16	6	0.364326574	b
17	7	0.599102556	b
18	8	0.659575412	b
19	9	0.634030849	b
20	10	0.717009826	b

“wide data”

	x	a	b
1	1	0.497868052	1.0775781
2	2	0.691927672	0.3341113
3	3	0.760313282	0.2227412
4	4	0.155401223	1.1391367
5	5	0.849457093	0.4929487
6	6	0.946817819	0.3643266
7	7	0.588419190	0.5991026
8	8	0.502250815	0.6595754
9	9	0.189779918	0.6340308
10	10	0.001836858	0.7170098

Several ways to switch between tidy and wide data!

gather() spread()
(from the “tidyverse” package which is included in the tidyverse)

Handle, re-organize and create data in R

- Use pipes (%>%) to:
 - Convert data from a “wide” format to a “long” format (tidy data)
 - Summarize data (similar to the Pivot function in Excel)
 - Create new data
 - Filter out data
 - Plot and visualize specific parts of your data (both raw and data summaries)
 - Sort your data
- Show how to organize your scripts in R using clickable headers (RStudio feature)



Handle, re-organize and create data in R

- If you want to learn more (which we hope after this session), have a look at these Cheat Sheets: <https://www.rstudio.com/resources/cheatsheets/>

Data Transformation with dplyr :: CHEAT SHEET

dplyr functions work with pipes and expect **tidy data**. In tidy data:

Summarise Cases

These apply **summary functions** to columns to create a new table of summary statistics. Summary functions take vectors as input and return one value (see back).

VARIATIONS

`summarise_all()` - Apply funs to every column.
`summarise_at()` - Apply funs to specific columns.
`summarise_if()` - Apply funs to all cols of one type.

Group Cases

Use `group_by()` to create a "grouped" copy of a table. dplyr functions will manipulate each "group" separately and then combine the results.

`group_by(data, ..., add = FALSE)`
 Returns copy of table grouped by ...
 $g_iris \leftarrow \text{group_by}(iris, Species)$

`ungroup(x, ...)`
 Returns ungrouped copy of table.
`ungroup(g, iris)`

Manipulate Cases

EXTRACT CASES

Row functions return a subset of rows as a new table.

Logical and boolean operators to use with filter()

< <= is.na() !is.na() | xor()
 > >= !is.na() !| &

See ?base::logical and ?Comparison for help.

ARRANGE CASES

ADD CASES

Manipulate Variables

EXTRACT VARIABLES

Column functions return a set of columns as a new vector or table.

Use these helpers with `select(...)`, e.g. `select(mtcars, starts_with("Sepo"))`
`contains(match)` `range(..., prefix, range)` ; e.g. `mppc:cycl`
`ends_with(match)` `one_of(...)` ; e.g. `Species`
`matches(match)` `starts_with(match)`

MAKE NEW VARIABLES

These apply **vectorized functions** to columns. Vectorized funs take vectors as input and return vectors of the same length as output (see back).

R Studio

RStudio® is a trademark of RStudio, Inc., & CC BY SA RStudio.com | info@rstudio.com | 844-448-1212 | rstudio.com | Learn more with browsable vignettes(package=c("dplyr", "tidyverse")) | dplyr 0.7.0 · tibble 1.2.0 · Updated: 2017-03-27

Data Visualization with ggplot2 : : CHEAT SHEET



Basics

ggplot2 is based on the grammar of graphics, the idea that you can build every graph from the same components: a **data set**, a **coordinate system**, and **geoms**—visual marks that represent data points.

DATA + GEOM = PLOT

To display values, map variables in the data to visual properties of the geom (aesthetics) like size, color, and x and y locations.

DATA + GEOM + COORDINATE SYSTEM = PLOT

Complete the template below to build a graph.

```
ggplot(data = DATA) +  
  COORDINATE FUNCTION(mapping = aes(MAPPINGS),  
  stat = STAT, position = POSITION) +  
  COORDINATE FUNCTION +  
  EACH FUNCTION +  
  SCALE FUNCTION +  
  THEME FUNCTION
```

ggplot(data = mpg, aes(x = cyl, y = hwy)) Begins a plot that you finish by adding layers to. Add one geom function per layer.

geom (**mapping**, **data**, **geom**)

ggplot(x ~ cyl, y ~ hwy, data = mpg, geom = "point")

Creates a complete plot with given data, geom, and mapping. Supplies many common defaults.

last, **plot** (**mpg**) Shows the last plot.

ggname (**plot**, **name**, **width** = 5, **height** = 5) Saves last plot as **5x5** file named **plot.png** in working directory. Matches file type to file extension.

Geoms

Use a geom function to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

GRAPHICAL PRIMITIVES

- b + geom_point(mapping = aes(x = long, y = lat))
- b + geom_rect(mapping = aes(x = long, y = lat))
- b + geom_text(mapping = aes(x = long, y = lat))
- b + geom_bar(mapping = aes(x = group))
- b + geom_line(mapping = aes(x = group, group, linetype, size))
- b + geom_polygon(mapping = aes(group = group))
- b + geom_rect(mapping = aes(xmin = min, xmax = max, ymin = min, ymax = max))
- b + geom_ribbon(mapping = aes(ymin = min, ymax = max, alpha = alpha, fill = fill, linetype, size))

LINE SEGMENTS

common aesthetics: x, y, alpha, color, linetype, size

- b + geom_abline(mapping = aes(intercept, slope = 1))
- b + geom_hline(mapping = aes(intercept))
- b + geom_segment(mapping = aes(x = start_x, xend = end_x, y = start_y, yend = end_y))
- b + geom_spoke(mapping = aes(radii = 1))

ONE VARIABLE

continuous y

- c + ggplot(mpg, aes(hwy)) + c2 + ggplot(mpg)
- c + geom_area(mapping = "hwy")
- c + geom_bar(mapping = "hwy")
- c + geom_boxplot(mapping = "hwy")
- c + geom_dotplot(mapping = "hwy")
- c + geom_hex(mapping = "hwy")
- c + geom_histogram(binwidth = 5) + c5
- c + geom_rug(mapping = "hwy")
- c + geom_violin(mapping = "hwy")

discrete x, continuous y

- c + geom_bar(mapping = aes(x = factor))
- c + geom_boxplot(mapping = aes(x = factor))
- c + geom_dotplot(mapping = "hwy")
- c + geom_hex(mapping = "hwy")
- c + geom_linerange(mapping = "hwy")
- c + geom_parallel(mapping = "hwy")

discrete x, discrete y

- c + geom_count(mapping = aes(x = factor, color = factor))
- c + geom_hex(mapping = aes(x = factor, y = factor))
- c + geom_raster(mapping = aes(x = factor, y = factor))

TWO VARIABLES

continuous x, continuous y

- c + geom_abline(mapping = aes(x = 1, y = 1, check_overlap = TRUE), x = 1, alpha, angle, color, family, fontface, hjust, intercept = 1, label = label, linecap = linecap, size = size)
- c + geom_ribbon(mapping = aes(x = 1, y = 1, width = 2), x = 1, alpha, color, fill, shape, size, stroke)
- c + geom_text(mapping = aes(x = 1, y = 1, width = 2), x = 1, alpha, color, fill, shape, size, weight)

discrete x, discrete y

- c + geom_hex(mapping = aes(x = factor, y = factor))
- c + geom_raster(mapping = aes(x = factor, y = factor))

continuous bivariate distribution

- b + geom_diamonds(mapping = aes(x = mean, y = mean, size = size))
- b + geom_hexbin(binwidth = 0.25, 0.50, 1.00, 2.00, 4.00, 8.00, 16.00, 32.00, 64.00, 128.00, 256.00, 512.00, 1024.00, 2048.00, 4096.00, 8192.00, 16384.00, 32768.00, 65536.00, 131072.00, 262144.00, 524288.00, 1048576.00, 2097152.00, 4194304.00, 8388608.00, 16777216.00, 33554432.00, 67108864.00, 134217728.00, 268435456.00, 536870912.00, 1073741824.00, 2147483648.00, 4294967296.00, 8589934592.00, 17179869184.00, 34359738368.00, 68719476736.00, 137438953472.00, 274877906944.00, 549755813888.00, 1099511627776.00, 2199023255552.00, 4398046511104.00, 8796093022208.00, 17592186044416.00, 35184372088832.00, 70368744177664.00, 140737488355328.00, 281474976710656.00, 562949953421312.00, 1125899906842640.00, 2251799813685280.00, 4503599627370560.00, 9007199254741120.00, 18014398509482240.00, 36028797018964480.00, 72057594037928960.00, 144115188075857920.00, 288230376151715840.00, 576460752303431680.00, 1152921504606863200.00, 2305843009213726400.00, 4611686018427452800.00, 9223372036854905600.00, 18446744073709811200.00, 36893488147419622400.00, 73786976294839244800.00, 147573952589678496000.00, 295147905179356992000.00, 590295810358713984000.00, 1180591620717427968000.00, 2361183241434855936000.00, 4722366482869711872000.00, 9444732965739423744000.00, 18889465931478847488000.00, 37778931862957694976000.00, 75557863725915389952000.00, 151115727451830779840000.00, 302231454903661559680000.00, 604462909807323119360000.00, 1208925819614646238720000.00, 2417851639229292477440000.00, 4835703278458584954880000.00, 967140655691716990960000.00, 1934281311383433981920000.00, 3868562622766867963840000.00, 7737125245533735927680000.00, 15474250491067471855360000.00, 30948500982134943710720000.00, 61897001964269887421440000.00, 123794003928539774842880000.00, 247588007857079549685760000.00, 495176015714159099371520000.00, 990352031428318198743040000.00, 1980704062856636397486080000.00, 3961408125713272794972160000.00, 7922816251426545589944320000.00, 15845632502853091179888640000.00, 31691265005706182359777280000.00, 63382530011412364719554560000.00, 12676506002282472943850920000.00, 25353012004564945887701840000.00, 50706024009129891775403680000.00, 101412048018259783550807360000.00, 202824096036519567101614720000.00, 405648192073038734203229440000.00, 811296384146077468406458880000.00, 162259276829215493681297760000.00, 324518553658430987362595200000.00, 649037107316861974725185600000.00, 1298074214633723949450371200000.00, 2596148429267447898900742400000.00, 5192296858534895797801484800000.00, 10384593717069791595602969600000.00, 20769187434139583191205939200000.00, 41538374868279166382411878400000.00, 83076749736558332764823756800000.00, 166153499473116665529647513600000.00, 332306998946233331059295027200000.00, 664613997892466662118590054400000.00, 1329227995784933324237850108800000.00, 2658455991569866648475700217600000.00, 5316911983139733296951400435200000.00, 10633823966279466593902800870400000.00, 21267647932558933187805601740800000.00, 42535295865117866375611203481600000.00, 85070591730235732751222406963200000.00, 170141183460471465502444813926400000.00, 340282366920942931004889627852800000.00, 680564733841885862009779255705600000.00, 1361129467683771724019558511411200000.00, 2722258935367543448039117022822400000.00, 5444517870735086896078234045644800000.00, 10889035741470173792156468091296000000.00, 21778071482940347584312936182592000000.00, 43556142965880695168625872365184000000.00, 87112285931761390337251744730368000000.00, 174224571863522780674503489460736000000.00, 348449143727045561349006978921472000000.00, 696898287454091122698013957842944000000.00, 1393796574908182245396027915685888000000.00, 2787593149816364490792055831371776000000.00, 5575186299632728981584111662743552000000.00, 1115037259926545796316822332548672000000.00, 2230074519853091592633644665097344000000.00, 4460149039706183185267289330194688000000.00, 8920298079412366370534578660389376000000.00, 17840596158824732741069157320778752000000.00, 3568119231764946548213831464155744000000.00, 7136238463529893096427662928311488000000.00, 1427247692705978619285532585662296000000.00, 2854495385411957238571065171324592000000.00, 5708990770823914477142130342649184000000.00, 1141798154164782895428426068529368000000.00, 2283596308329565790856852137058736000000.00, 4567192616659131581713704274117472000000.00, 9134385233318263163427408548234944000000.00, 1826877046663652632685481709646988000000.00, 3653754093327305265370963419293976000000.00, 7307508186654610530741926838587952000000.00, 1461501637330922106148385367717584000000.00, 2923003274661844212296770735435168000000.00, 5846006549323688424593541470870336000000.00, 11692013098647376849187082941740672000000.00, 23384026197294753698374165883481344000000.00, 46768052394589507396748331766962688000000.00, 9353610478917901479349666353392536000000.00, 18707220957835802958699332706785072000000.00, 37414441915671605917398665413570144000000.00, 74828883831343211834797330827140288000000.00, 14965776766268642366959466165428056000000.00, 29931553532537284733918932330856112000000.00, 59863107065074569467837864661712224000000.00, 119726214130149138935655329323424448000000.00, 239452428260298277871310658646848896000000.00, 478904856520596555742621317293697792000000.00, 957809713041193111485242634587395584000000.00, 1915619426082386222910485269174791168000000.00, 3831238852164772445820970538349582336000000.00, 7662477704329544891641941076698964672000000.00, 15324955408659089783283882153397929344000000.00, 30649910817318179566567764306795858688000000.00, 61299821634636359133135528613591717376000000.00, 12259964326927271826627105722718343552000000.00, 24519928653854543653254211445436687064000000.00, 49039857307709087306508422890873374128000000.00, 98079714615418174613016845781746748256000000.00, 196159429230836349226033691563493496512000000.00, 392318858461672698452067383126986992024000000.00, 784637716923345396904134766253973984048000000.00, 1569275433846690793808269532507947968096000000.00, 3138550867693381587616539065015895936192000000.00, 6277101735386763175233078130031791872384000000.00, 12554203470773526350466156260063583744768000000.00, 25108406941547052700932312520127167489536000000.00, 5021681388309410540186462504025433497872000000.00, 10043362776618821080372925008050866995744000000.00, 20086725553237642160745850016101733991488000000.00, 40173451106475284321491700032203467982976000000.00, 80346902212950568642983400064406935965952000000.00, 16069380442580113728596680012881387193184000000.00, 32138760885160227457193360025762754386368000000.00, 64277521770320454914386720051525508772736000000.00, 128555043540640909828773440103051017545472000000.00, 257110087081281819657546880206102035090944000000.00, 514220174162563639315093760412204070181888000000.00, 1028440348325127278630187520824408140363776000000.00, 2056880696650254557260375041648816280727552000000.00, 4113761393300509114520750083297632561455088000000.00, 8227522786601018229041500166595265122910176000000.00, 16455045573002036458083003331985312558202352000000.00, 32910091146004072916166006663970625116404704000000.00, 65820182292008145832332001337941250232809408000000.00, 131640364584016291664664002675882504656048816000000.00, 263280729168032583329328005351765009312097632000000.00, 526561458336065166658656010703530018624195264000000.00, 1053122916672130333317120214067650372488390528000000.00, 2106245833344260666634240428135307549776781056000000.00, 4212491666688521333268480856270615099553562112000000.00, 8424983333377042666536961712541230199107124224000000.00, 1684996666675408533307382342508260398214248448000000.00, 3369993333350817066614764685016520796428976992000000.00, 6739986666701634133229529370033041592857539984000000.00, 13479973333403268266458858740066083185715199768000000.00, 26959946666806536532917717480132166371430399536000000.00, 53819893333613073065835434960264332742867990672000000.00, 10763978666726544613167067992052866549575981344000000.00, 21527957333453089226334135984105733099151962688000000.00, 43055914666906178452668267968211466198303925376000000.00, 86111829333812356905336535936422932396607850752000000.00, 17222365866762471381067307187285864679321571504000000.00, 34444731733524942762134614374571730358643143008000000.00, 68889463467049885524269228749143460717286286016000000.00, 13777892693409977104853845749828692143457572032000000.00, 27555785386819954209687691499657384286915144064000000.00, 55111570773639908419375382999314768573830288128000000.00, 11022314154727951683875676599862937148766056256000000.00, 22044628309455903367751353199725874295532112512000000.00, 4408925661891180673550270639945174859106224024000000.00, 8817851323782361347100541279890349718212448048000000.00, 1763570266764682674220108255978069843642489696000000.00, 3527140533529365348440216511956139687284979392000000.00, 7054281067058730696880433023872279374569958784000000.00, 14108562134117461393760866047544587549139915768000000.00, 28217124268234922787521732095089155098279831536000000.00, 56434248536469845575043464190178310196559663072000000.00, 112868491072939691150086928380356620393119326144000000.00, 225736982145879382300173856760713240786238652288000000.00, 451473964291758764600347713521426481572477305576000000.00, 902947928583517529200695427042852963149554611152000000.00, 1805895858567035058401390854085715926299109222304000000.00, 361178171713407011680278170817143185258821844460800000.00, 722356343426814023360556341634286370517643688921600000.00, 144471268685362804672111268326573744103528777784320000.00, 288942537370725609344222536653147488207057555568640000.00, 577885074741451218688445073306294976414115111137280000.00, 115577014942902423737689146661258995282823022274456000.00, 231154029885804847475378293322519990565646044548912000.00, 462308059771609694950756586645039981131292089097824000.00, 924616119543219389901513173290079962262584178195648000.00, 184923223858643877980302634658015992452516835639136000.00, 369846447717287755960605269316031984905033671138272000.00, 73969289543457551192121023863206398880006734276544000.00, 147938570886911102384241047726413977760013468553088000.00, 295877141773822204768482095452831555200269377067776000.00, 591754283547644409536964190905663110405338754135552000.00, 118350895309528881907392838181132622081077508227104000.00, 236701780619057763814785676362265244162155016542088000.00, 473403561238115527629571352724530488324310033084176000.00, 946807122476231055259142705449060976648620066168352000.00, 1893614244952462110518285410898121953297240132336704000.00, 3787228489904924221036570821796243906594480264674148000.00, 7574456979809848442073141643592487813188960529348296000.00, 1514891395801969688414628328718495662637792105868579200.00, 3029782791603939376829256657436991325355584211737158400.00, 6059565583207878753658513314873882650711168423474316800.00, 12119131166415757507377

- Data Import Cheat Sheet
 - Base R Cheat Sheet
 - Advanced R Cheat Sheet
 - R Markdown Cheat Sheet

and many many more..

KUL data from the biotest basin loaded from an URL

Database for Coastal Fish - KUL

LAST CHANGED: 22 NOVEMBER 2017

The database KUL, with data from fishing with net and fyke net in coastal Waters, has been in use since year 2006 and provides quality assured catch data of coastal fish. The base also store information such as individual gender, length, weight and age. The database complements the internal database FiRRe, which stores data collected since the 1960s. Continuous work is done to transfer older data to the database KUL. The database is in Swedish.



Lets work with some KUL fish data!

