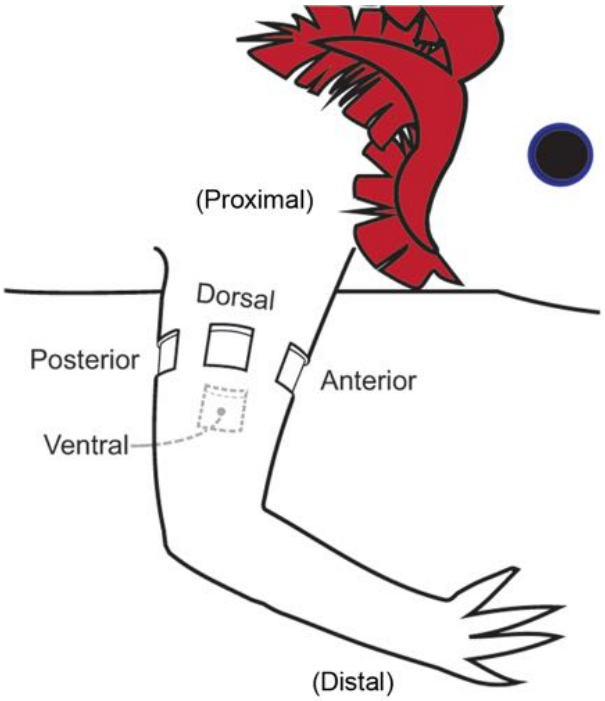
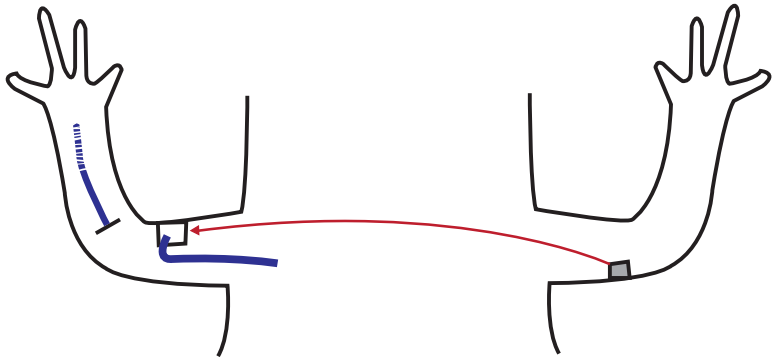
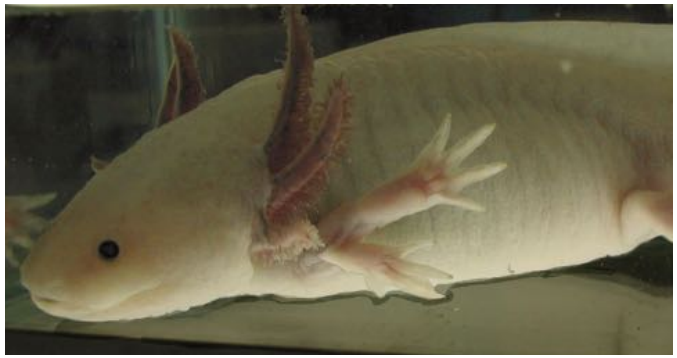
Data Cleaning & Tidy Data

Requirements for Amphibian Limb Regeneration

1. Generate a wound epithelium
2. Innervate the wound epithelium
3. Establish a positional disparity







(Endo *Dev Bio,* 2004)

Example image of a whole mount cartilage/bone stain on a limb where I performed the ALM surgery

Ectopic limb formed in
Site of ALM surgery… cool/weird huh?!

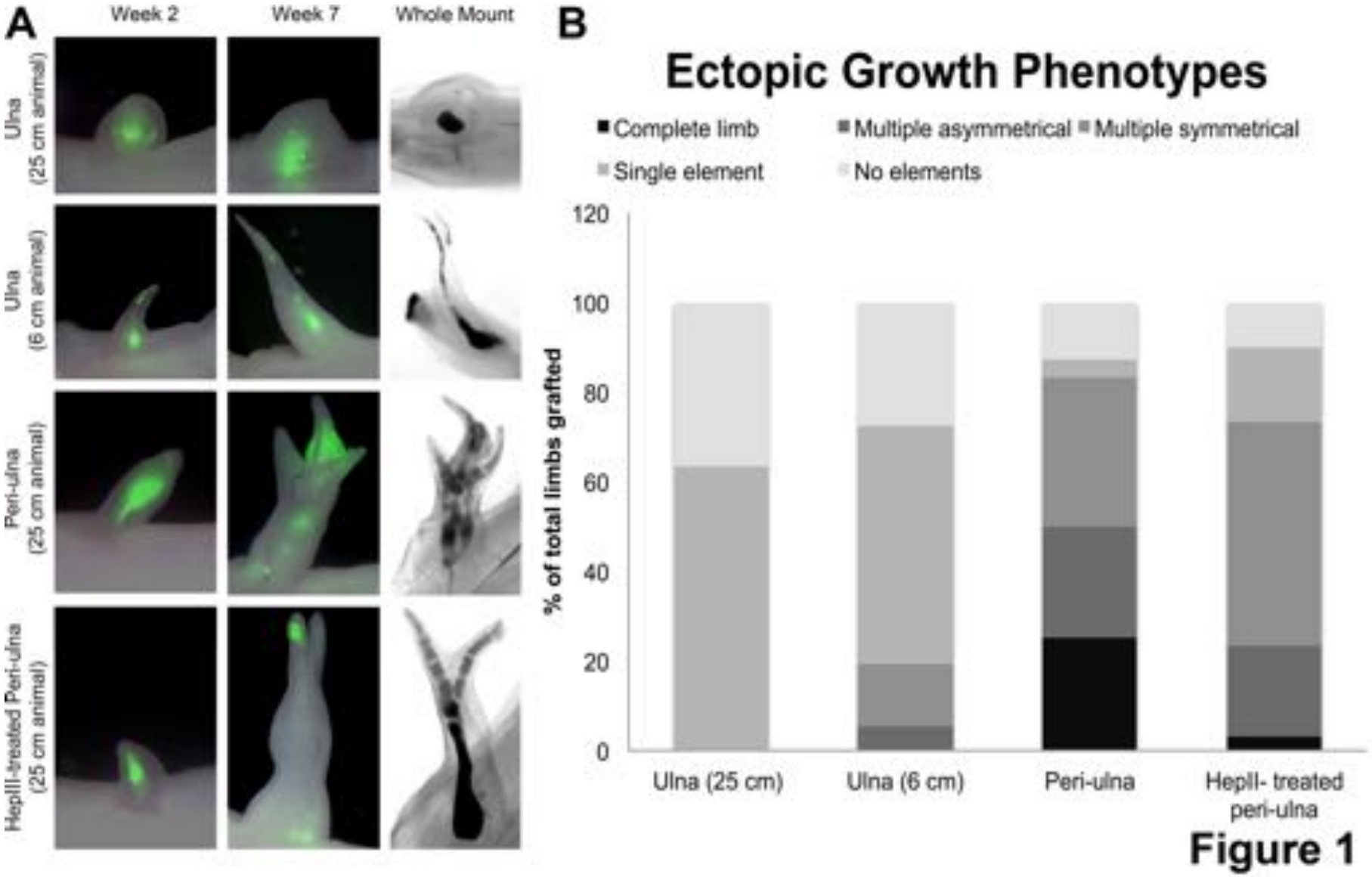

A
Week 2
Week 7
Whole Mount
Ulna (25 cm animal)
Ulna (6 cm animal)
Peri-ulna (25 cm animal)
HepIII-treated Peri-ulna (25 cm animal)
B
Ectopic Growth Phenotypes
Complete limb
Multiple asymmetrical
Multiple symmetrical
Single element
No elements
% of total limbs grafted
120
100
80
60
40
20
0
Ulna (25 cm)
Ulna (6 cm)
Peri-ulna
HepII- treated peri-ulna

Figure 1

Loading and initial chopping

- What is the optimal way to load the data and get row titles?
- Chop extra rows/cols
- Standardize text
- Find bad rows using regular expressions

Beyond == for Strings

How do I match which strings have an e in them?

```
c(“hello”, “goodbye”, “salutations”)
```

Beyond == for Strings

```
> hello <- c("hello", "goodbye", "salutations")


> grep("e", hello)
[1] 1 2
```

Beyond == for Strings

```
> hello <- c("hello", "goodbye", "salutations")
```

```
> grepl("e", hello)
[1] TRUE TRUE FALSE
```

Exercise - which of these have the letter l in them?

Matching Patters with REGEXP

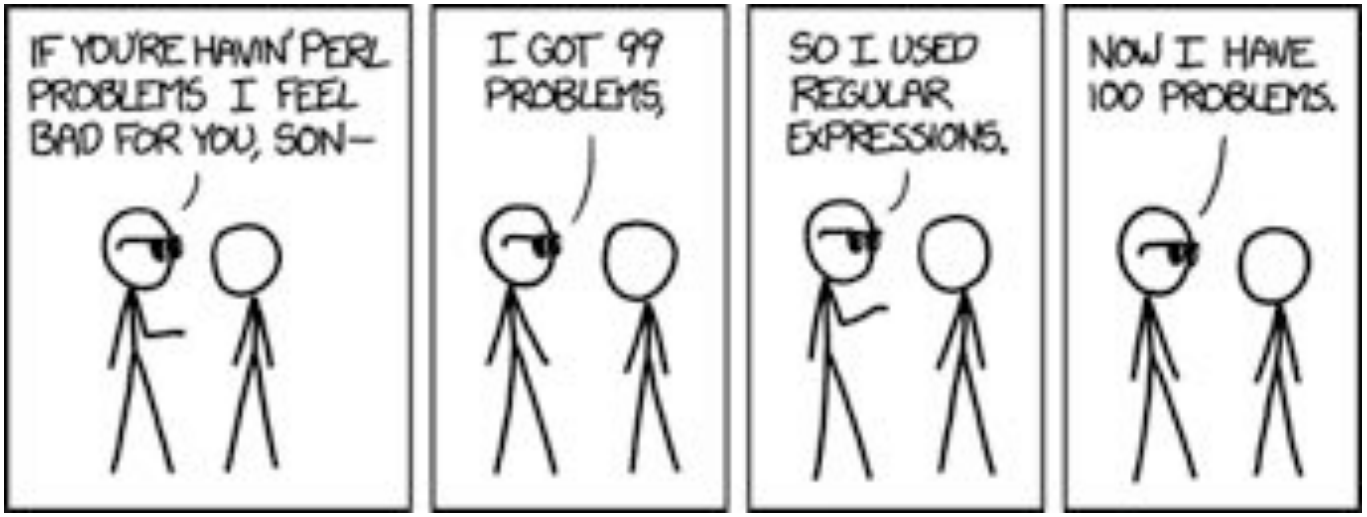
Which of these have an “o” not at the end of the line?

```
c(“hello”, “goodbye”, “salutations”)
```

```
grepl(".o.", hello)
```

Whither the '.'

- . matches any character
- It is a metacharacter – there are many!



xkcd

Some General Regexps

- . - any character apart from a newline.
- \\d - any digit.
- \\s - any whitespace (space, tab, newline).
- \n – a line break
- [abc] - match a, b, or c.
- [!abc] - match anything except a, b, or c.

Which of these have a digit-space combo?

```
c(“33 points”, “hello kitty”, “drove up 95”)
```

Line start and End

Which of these have an “o” not at the end of the line?

```
c(“hello”, “goodbye”, “salutations”)
```

```
!grepl(”o$", hello)
```

^ - Start of line, $ - end of line

But what if I want to find $

- Escape characters allow you to ignore special meaning of $, ^, or other metacharacters
- \\$ finds $
- \\\ finds \
- \\(finds (

Multiple Letters

Which of these have a double l?

```
c(“hello”, “goodbye”, “salutations”)
```

```
grepl(”ll", hello)

grepl(”l{2,}", hello)
```

Modifiers for Multiples

- ? - 0 or 1
- + - 1 or more
- * - 0 or more
- {n} - exactly n
- {n,} - n or more
- {,m} - at most m
- {n,m} - between n and m

Exercise

- Which of these have 1 number?
- Which have 2 numbers?
- Which have >2 numbers?

```
c("green 5", "blue 32", "blue
32", "hut hut", "2348923”)
```

Stump Us!

- Using http://bit.ly/regexp-cheat come up with a devilish regexp problem for us

- When you've got it, share it, and we'll try and solve it

Regexp and Filter

How would you use Regexps to filter out bad rows in the Axoltl data?

Substitution

- Regexps can be most useful in changing text

Pattern **Replacement**

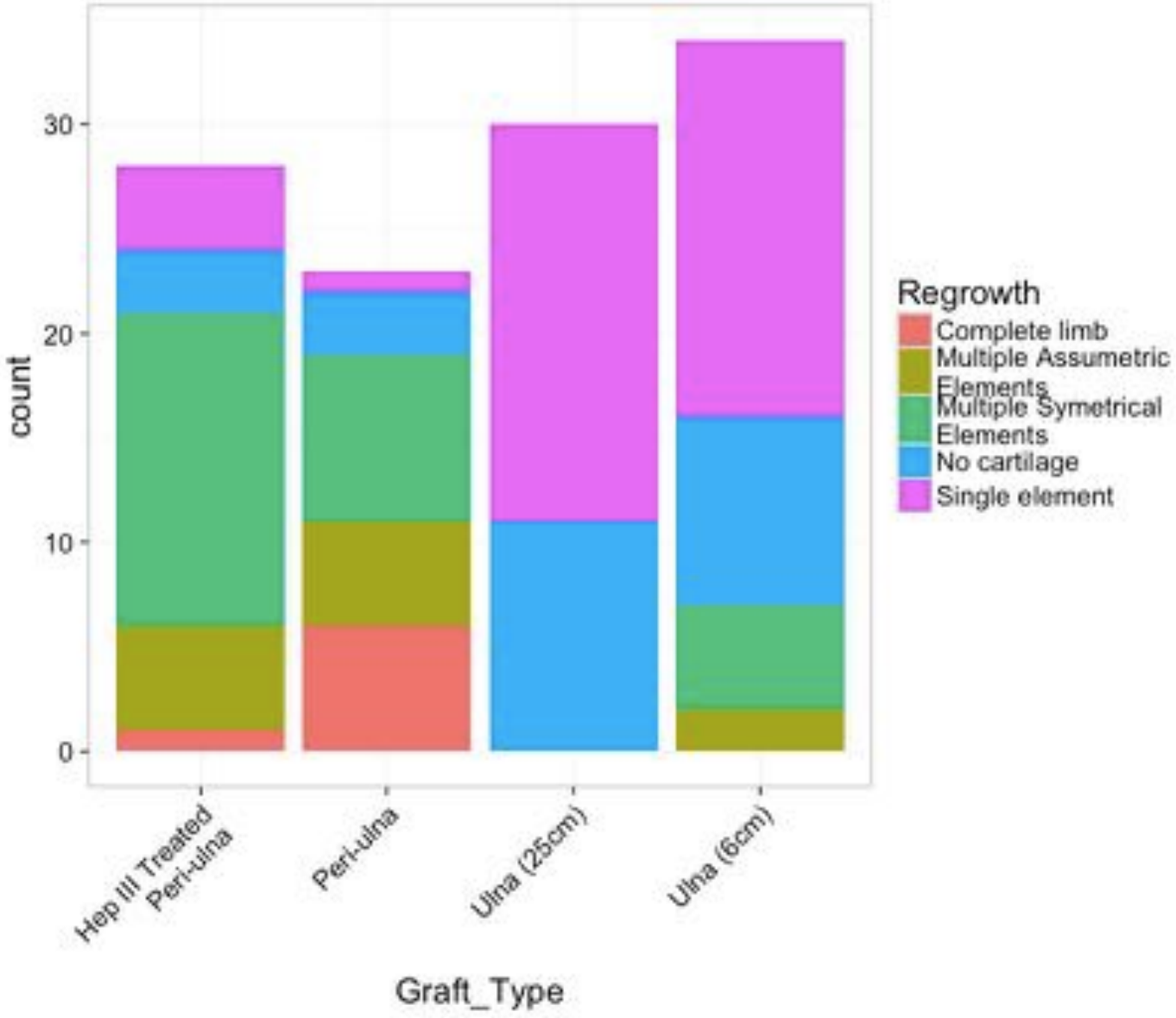
```
> gsub("hello", "ciao", hello)

[1] "ciao"          "goodbye"      "salutations"
```

Exercise

Use Substitution to clear out values from cells that have “completed” text in them

Use Substitution to Simplify Text



Extracting Pieces

```
> gsub("^(\\w*) (.*)",
       "\\1",
       "Are you a string?")
```

Just the first ()

```
[1] "Are"
```

Extracting Pieces

```
> gsub("^(\\w*) (.*)",
       "\\2",
       "Are you a string?")


[1] "you a string?"
```

Fill down

```
> adf <- data.frame(x = c(1,NA, NA, NA, 3, NA,
NA, NA))

> adf
   x
1  1
2 NA
3 NA
4 NA
5  3
6 NA
7 NA
8 NA
```

Fill down

```
library(tidyr)

adf %>%
   fill(x, .direction="down")
```

```
  x
1 1
2 1
3 1
4 1
5 3
6 3
7 3
8 3
```

Try this with the EXPERIMENT column