Bio Stats II : Lab 3

Acknowledgements: Punt & Branch Labs (U Washington)

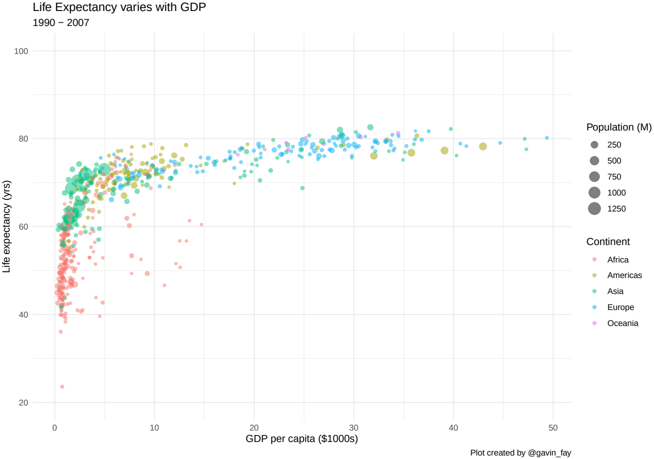
Gavin Fay

01/31/2023

Lab schedule

1/18: Introduction to R and R Studio, working with data
1/25: Intro to Visualization
1/31: Probability, linear modeling
2/08: Data wrangling, model summaries
2/15: Iteration
2/22: Creating functions, debugging
3/01: Simulation, Resampling
3/15: Flex: more modeling (brms, glmmTMB)
3/29: Spatial data or tidymodeling

Review


Life Expectancy varies with GDP
1990 − 2007
Life expectancy (yrs)
100
80
60
40
20
0
10
20
30
40
50
GDP per capita ($1000s)
Population (M)
250
500
750
1000
1250
Continent
Africa
Americas
Asia
Europe
Oceania
Plot created by @gavin_fay

```
> suppressPackageStartupMessages(library(tidyverse))
> library(gapminder)
> gapdata <- gapminder %>%
+   filter(year >= 1990 & year <= 2007) %>%
+   group_by(continent)
> ggplot(gapdata, aes(x = gdpPercap/1000,
+                     y = lifeExp,
+                     size = pop/1000000,
+                     color = continent)) +
+   geom_point(alpha = 0.5) +
+   ylim(20,100) +
+   labs(title = "Life Expectancy varies with GDP",
+          subtitle = "1990 - 2007",
+           x = "GDP per capita ($1000s)",
+           y = "Life expectancy (yrs)",
+          caption = "Plot created by @gavin_fay",
+        size = "Population (M)",
+        color = "Continent") +
+   theme_minimal()
```

Recommended reading

An introduction to R (Venables et al.)
– http://cran.r-project.org/doc/manuals/R-intro.pdf - Today's material: Chapters 8, 11.

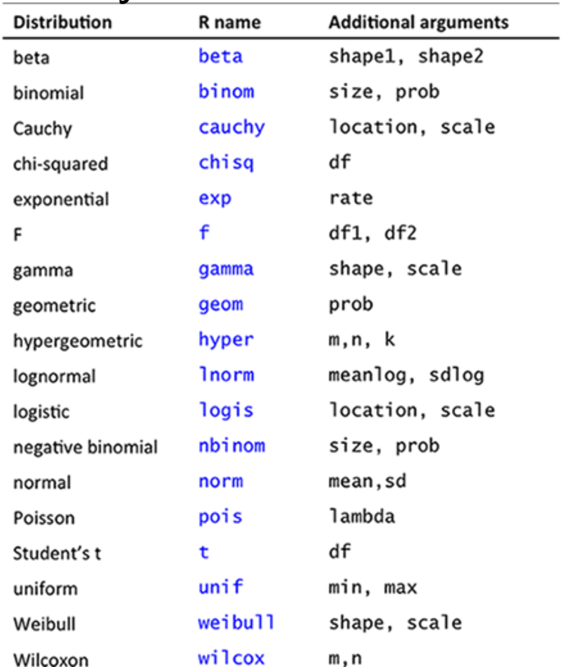
Probability distributions in `R`

`R` includes a set of probability distributions that can be used to simulate and model data.

If the function for the probability model is named `xxx`

- `pxxx`: the cumulative distribution $P(X \leq x)$
- `dxxx`: the probability distribution/density function $f(x)$
- `qxxx`: the quantile q, the smallest x such that $P(X \leq x) > q$
- `rxxx`: generate a random variable from the model `xxx`

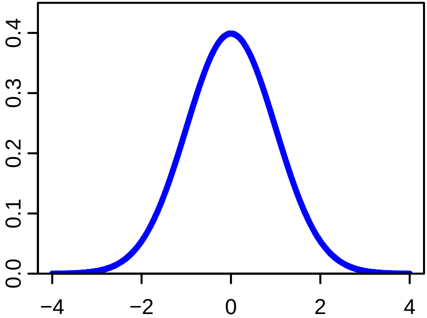
Probability distributions in R



Distribution	R name	Additional arguments
beta	beta	shape1, shape2
binomial	binom	size, prob
Cauchy	cauchy	location, scale
chi-squared	chisq	df
exponential	exp	rate
F	f	df1, df2
gamma	gamma	shape, scale
geometric	geom	prob
hypergeometric	hyper	m,n, k
lognormal	lnorm	meanlog, sdlog
logistic	logis	location, scale
negative binomial	nbinom	size, prob
normal	norm	mean,sd
Poisson	pois	lambda
Student's t	t	df
uniform	unif	min, max
Weibull	weibull	shape, scale
Wilcoxon	wilcox	m,n

Standard normal distribution

$(\mu = 0,\ \sigma^2 = 1)$



Functions for normal distribution

Values of x for different quantiles

`qnorm(p, mean=0, sd=1, lower.tail=TRUE, log.p=FALSE)`

```
> quants <- qnorm(c(0.01,0.025,0.05,0.95,0.975,0.99))
> round(quants,2)
[1] -2.33 -1.96 -1.64  1.64  1.96  2.33
```

$P(X \leq x)$

`pnorm(q, mean=0, sd=1, lower.tail=TRUE, log.p=FALSE)`

```
> pnorm(quants)
[1] 0.010 0.025 0.050 0.950 0.975 0.990
```

Functions for normal distribution

Density (probability ‘mass’ per unit value of x)

```
> dnorm(quants, mean = 0, sd = 1)
[1] 0.02665214 0.05844507 0.10313564 0.10313564 0.05844507
```

Generating standard normal random variables

```
> rnorm(n=10, mean = 0, sd = 1)
 [1]  1.3149588  0.9781675  0.8817912  0.4822047  0.9657529
 [7]  0.2839578 -0.1616986  1.9355718  1.7232308
```

Generating random numbers

Often a good idea to use `set.seed()` and save the script detailing which number was used.

This ensures you can exactly repeat your results.

```
> set.seed(42)
> rnorm(3)
[1]  1.3709584 -0.5646982  0.3631284
> rnorm(3)
[1]  0.6328626  0.4042683 -0.1061245
> set.seed(42)
> rnorm(3)
[1]  1.3709584 -0.5646982  0.3631284
> rnorm(3)
[1]  0.6328626  0.4042683 -0.1061245
```

The `sample()` function

To generate random numbers from discrete sets of values:
- With or without replacement
- Equal or weighted probability

Extremely useful function that underlies many modern statistical techniques:
- Resampling
- Bootstrapping
- Markov-chain Monte-Carlo (MCMC)

e.g. Roll 10 dice

```
> sample(1:6, size = 10, replace = TRUE)
 [1] 4 1 5 6 4 2 2 3 1 1
```

Pick 3 species of bear

```
> bears <- c("American Black", "Asian Black", "Brown",
+             "Panda", "Polar", "Sloth", "Spectacled", "Sun")
> sample(bears, size = 3, replace = FALSE)
[1] "Brown" "Panda" "Polar"
```

Lab Exercise 1/3

a. Generate 1,000 random normal numbers with mean 24 and standard deviation 10. Find the proportion of those random numbers that are ≥ 2 standard deviations from the sample mean.

b. Flip a (fair) coin six times.

c. Find the probability of getting six heads on those six flips (i.e. $P(X = 6)$ given $n = 6$).

d. How much more likely is it to get three heads than six?

e. For a standard normal random variable, find the number x such that $P(-x \leq X \leq x) = 0.24$.

f. The mean rate of arrival of alewives at a weir is 3.5 per hour. Plot the probability distribution function for the number of alewife arrivals in an hour.

g. Find the 95% confidence interval for the number of alewives arriving per day.

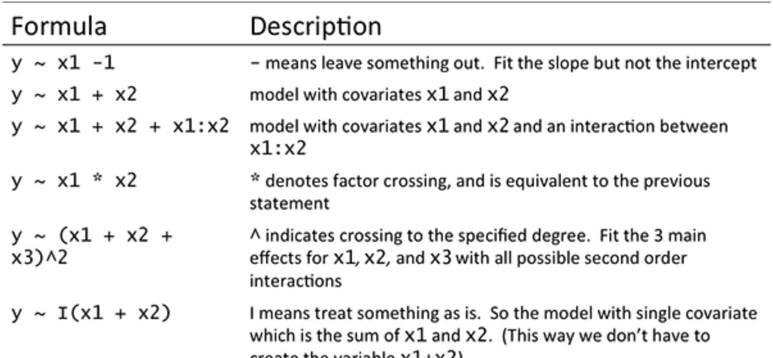
Linear models in R

Recall linear regression model:

$$y_i = \sum_{j=0}^{p} \beta_j x_{ij} + \varepsilon_i \ , \ \varepsilon_i \sim N(0, \sigma^2) \ , \ i = 1, \ldots, n$$

In matrix form: $y = X\beta + \varepsilon$

Model formulae in `R`, $y \sim x$, try `?formula`



Formula	Description
`y ~ x1 -1`	– means leave something out. Fit the slope but not the intercept
`y ~ x1 + x2`	model with covariates x1 and x2
`y ~ x1 + x2 + x1:x2`	model with covariates x1 and x2 and an interaction between x1:x2
`y ~ x1 * x2`	* denotes factor crossing, and is equivalent to the previous statement
`y ~ (x1 + x2 + x3)^2`	^ indicates crossing to the specified degree. Fit the 3 main effects for x1, x2, and x3 with all possible second order interactions
`y ~ I(x1 + x2)`	I means treat something as is. So the model with single covariate which is the sum of x1 and x2. (This way we don't have to create the variable x1+x2)

Linear models, `lm()`

The basic function for fitting ordinary multiple models is `lm()`

`fitted.model <- lm(formula, data = data.frame)`

e.g. species richness on beaches (Zuur Chapters 5 & 27)

```
> RIKZ <- read_table(file = "../data/RIKZ.txt")
> RIKZ <- RIKZ |>
+   mutate(Richness = rowSums(RIKZ[,2:76]>0)) |>
+   select(Richness, 77:89)
> RIKZ_lm1 <- lm(Richness ~ NAP, data = RIKZ)
```

Extracting model information

The value of `lm()` is a fitted model object.
- a list of results of class `lm`.

Information about the fitted model can be extracted, displayed, plotted, using some generic functions, inlcuding:

- `anova(object1,object2)` Compares a submodel with an outer model and produces an analysis of variance table
- `coef(object)` Extract the regression coefficient
- `deviance(object)` Residual sum of squares
- `formula(object)` Extract the model formula
- `plot(object)` Produce four plots, showing residuals, fitted values and some diagnostics
- `predict(object, newdata=data.frame)` Model predictions on new data
- `residuals(object)` or `resid(object)` Extract the matrix of residuals
- `step(object)` forward or backward model selection using AIC
- `summary(object)` Print a comprehensive summary of the results
- `vcov(object)` Return variance-covariance matrix of main parameters

Summary objects

```
summary(lm1)
```

```
Call:
lm(formula = Richness ~ NAP, data = RIKZ)

Residuals:
    Min      1Q  Median      3Q     Max
-5.0675 -2.7607 -0.8029  1.3534 13.8723

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)   6.6857     0.6578  10.164 5.25e-13 ***
NAP          -2.8669     0.6307  -4.545 4.42e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.16 on 43 degrees of freedom
Multiple R-squared:  0.3245,    Adjusted R-squared:  0.3088
F-statistic: 20.66 on 1 and 43 DF,  p-value: 4.418e-05
```

broom helper functions

broom::tidy()

```
> library(broom)
> tidy(RIKZ_lm1, conf.int = TRUE)
# A tibble: 2 x 7
  term          estimate std.error statistic  p.value conf.low con
  <chr>            <dbl>     <dbl>     <dbl>    <dbl>    <dbl>
1 (Intercept)       6.69     0.658     10.2  5.25e-13     5.36
2 NAP              -2.87     0.631     -4.55 4.42e- 5    -4.14
```

broom::glance()

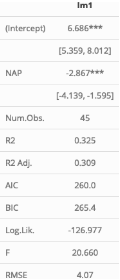
```
> glance(RIKZ_lm1)
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic   p.value    df logLik
      <dbl>         <dbl> <dbl>     <dbl>     <dbl> <dbl>  <dbl>
1     0.325         0.309  4.16      20.7 0.0000442     1  -127.
# ... with 3 more variables: deviance <dbl>, df.residual <int>,
```

{modelsummary}

The {modelsummary} package provides a wealth of options for creating publication-quality summary tables from fitted model objects, with bindings for most of the types of models we are using in this class.

There are lots of options. . . .

```
> library(modelsummary)
> modelsummary(list(lm1 = RIKZ_lm1),
+              statistic = "conf.int",
+              stars = TRUE)
```



	lm1
(Intercept)	6.686***
	[5.359, 8.012]
NAP	-2.867***
	[-4.139, -1.595]
Num.Obs.	45
R2	0.325
R2 Adj.	0.309
AIC	260.0
BIC	265.4
Log.Lik.	-126.977
F	20.660
RMSE	4.07

Are model assumptions met?

In the MASS library there are many functions.

Are samples independent? (Sample design.)

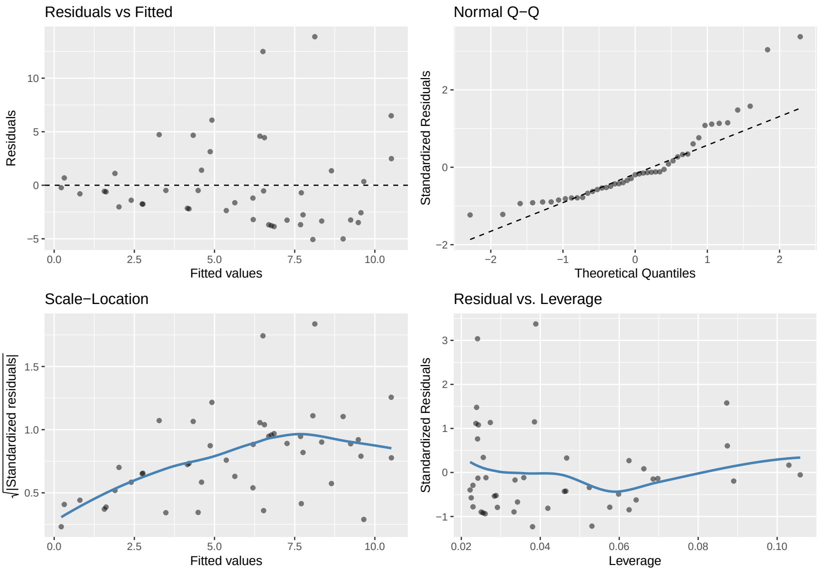
Residuals normally distributed?
- Histograms, qq-plots: qqplot() and qqline()
- Kolmogorov-Smirnov normality test: ks.test()
- Shapiro-Wilk normality test: shapiro.test()

Similar variance among samples?
- Boxplots
- Bartlett's test for equal variance: bartlett.test()
- Fligner-Killeen test for equal variance: fligner.test()

Checking assumptions

Model assumptions can be evaluated by plotting the model object.

```
> plot(RIKZ_lm1)
> # OR
> library(gglm)
> gglm(RIKZ_lm1)
> gglm(RIKZ_lm1)
```



Residuals vs Fitted

Normal Q−Q

Scale−Location

Residual vs. Leverage

Extracting model fit & residuals

`broom::augment()`

```
> RIKZ_lm1_aug <- augment(RIKZ_lm1, se_fit = )
> glimpse(RIKZ_lm1_aug)
Rows: 45
Columns: 8
$ Richness   <dbl> 11, 10, 13, 11, 10, 8, 9, 8, 19, 17, 6,
$ NAP        <dbl> 0.045, -1.036, -1.336, 0.616, -0.684, 1.
$ .fitted    <dbl> 6.556653, 9.655722, 10.515778, 4.919680,
$ .resid     <dbl> 4.4433465, 0.3442783, 2.4842223, 6.08031
$ .hat       <dbl> 0.02432839, 0.06623475, 0.08738853, 0.02
$ .sigma     <dbl> 4.151533, 4.208801, 4.189991, 4.100641,
$ .cooksd    <dbl> 0.0145788938, 0.0002601525, 0.0187094818
$ .std.resid <dbl> 1.08136537, 0.08564557, 0.62511744, 1.47
```

Extracting model fit & residuals

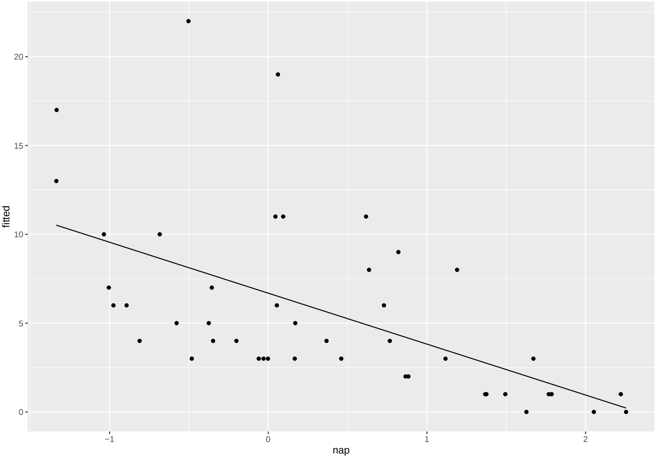
```
> RIKZ_lm1_aug <- augment(RIKZ_lm1,
+                         se_fit = TRUE)
> glimpse(RIKZ_lm1_aug)
Rows: 45
Columns: 9
$ Richness   <dbl> 11, 10, 13, 11, 10, 8, 9, 8, 19, 17, 6,
$ NAP        <dbl> 0.045, -1.036, -1.336, 0.616, -0.684, 1.
$ .fitted    <dbl> 6.556653, 9.655722, 10.515778, 4.919680,
$ .se.fit    <dbl> 0.6488474, 1.0706040, 1.2297395, 0.64280
$ .resid     <dbl> 4.4433465, 0.3442783, 2.4842223, 6.08031
$ .hat       <dbl> 0.02432839, 0.06623475, 0.08738853, 0.02
$ .sigma     <dbl> 4.151533, 4.208801, 4.189991, 4.100641,
$ .cooksd    <dbl> 0.0145788938, 0.0002601525, 0.01870948l8
$ .std.resid <dbl> 1.08136537, 0.08564557, 0.62511744, 1.47
```

Extracting model fit & residuals

```
> RIKZ_lm1_aug <- augment(RIKZ_lm1,
+                         interval = "confidence")
> glimpse(RIKZ_lm1_aug)
Rows: 45
Columns: 10
$ Richness   <dbl> 11, 10, 13, 11, 10, 8, 9, 8, 19, 17, 6,
$ NAP        <dbl> 0.045, -1.036, -1.336, 0.616, -0.684, 1.
$ .fitted    <dbl> 6.556653, 9.655722, 10.515778, 4.919680,
$ .lower     <dbl> 5.24812804, 7.49664302, 8.03577164, 3.62
$ .upper     <dbl> 7.865179, 11.814800, 12.995784, 6.216014
$ .resid     <dbl> 4.4433465, 0.3442783, 2.4842223, 6.08031
$ .hat       <dbl> 0.02432839, 0.06623475, 0.08738853, 0.02
$ .sigma     <dbl> 4.151533, 4.208801, 4.189991, 4.100641,
$ .cooksd    <dbl> 0.0145788938, 0.0002601525, 0.0187094818
$ .std.resid <dbl> 1.08136537, 0.08564557, 0.62511744, 1.47
```

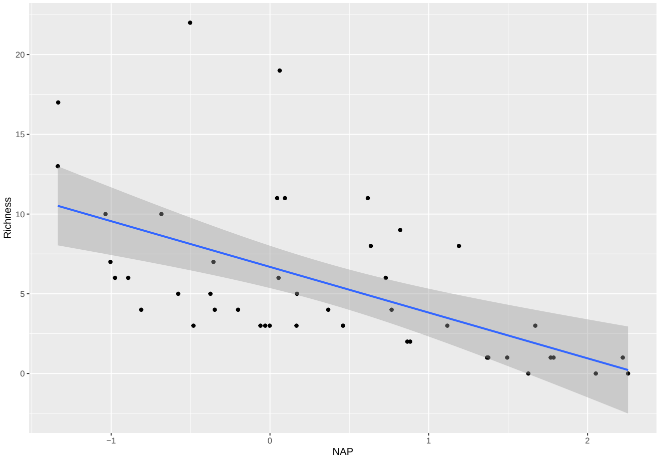
Extracting model fit & residuals

```
> RIKZ_lm1_aug |>
+   janitor::clean_names() |>
+   ggplot() +
+   aes(x = nap, y = fitted) +
+   geom_line() +
+   geom_point(aes(x = nap, y= richness))
```



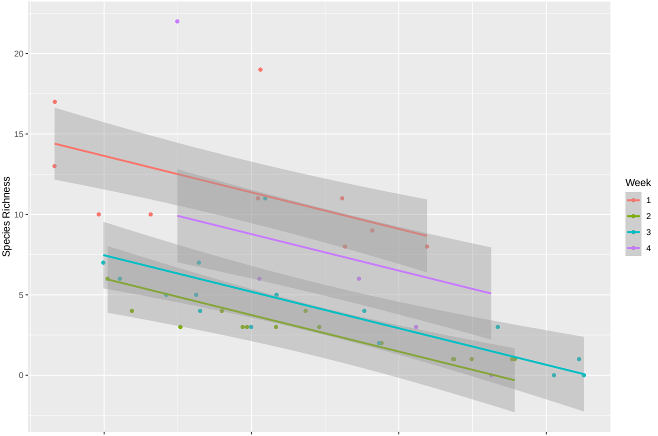
Fitting models directly on plots

```
> ggplot(RIKZ, aes(x = NAP, y = Richness)) +
+   geom_point() +
+   geom_smooth(method = "lm")
`geom_smooth()` using formula 'y ~ x'
```



Building and Comparing models

```
> library(moderndive)
> ggplot(RIKZ) +
+   aes(x = NAP, y = Richness, color = factor(week)) +
+   geom_point() +
+   labs(x = "NAP", y = "Species Richness",
+        color = "Week") +
+   geom_parallel_slopes(se = TRUE)
```



Building and Comparing models

An alternative model for the RIKZ species richness is:

```
> RIKZ_lm2 <- lm(Richness ~ NAP + factor(week), data = RIKZ)
> #Can also create this model using the `update()` function:
> RIKZ_lm2 <- update(RIKZ_lm1, .~. + factor(week))
```

Compare models with `AIC()` and via `anova()`

```
> anova(RIKZ_lm1, RIKZ_lm2)
Analysis of Variance Table

Model 1: Richness ~ NAP
Model 2: Richness ~ NAP + factor(week)
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     43 744.12
2     40 357.00  3    387.11 14.458 1.581e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Lab exercise 2/3

1. Extract the residuals from the RIKZ_lm2 model
2. Are the linear regression assumptions met? Explain your reasoning
3. Summarize the results of the model. What are the parameter estimates telling you about species richness on these beaches?

Model predictions for new data

Both augment() and the predict() function can be used to obtain predictions from a fitted model object to a new data frame.

This can be useful when:

- a subset of the data was reserved from the fitting process (‘test data’)
- you want to obtain model predictions at values other than the data (for example when plotting fitted values)

```
> newdata <- RIKZ |>
+   slice(1:10) |>
+   mutate(NAP = NAP + 1)
> new_predictions <- augment(RIKZ_lm1, newdata = newdata)
```

More on interrogating model results

You can also interrogate model objects directly. Type `names(object)` to get a list of the components of `object`.

```
> names(RIKZ_lm1)
 [1] "coefficients"  "residuals"     "effects"       "rank"
 [5] "fitted.values" "assign"        "qr"            "df.re
 [9] "xlevels"       "call"          "terms"         "model
```

`str()` can also be used.

More on interrogating model results

Note that summary(object) is also a list, and components can be extracted from here too.

```
> names(summary(RIKZ_lm1))
 [1] "call"          "terms"         "residuals"     "coeff
 [5] "aliased"       "sigma"         "df"            "r.squ
 [9] "adj.r.squared" "fstatistic"    "cov.unscaled"
> str(summary(RIKZ_lm1))
List of 11
 $ call         : language lm(formula = Richness ~ NAP, dat
 $ terms        :Classes 'terms', 'formula'  language Richn
  .. ..- attr(*, "variables")= language list(Richness, NAP)
  .. ..- attr(*, "factors")= int [1:2, 1] 0 1
  .. .. ..- attr(*, "dimnames")=List of 2
  .. .. .. ..$ : chr [1:2] "Richness" "NAP"
  .. .. .. ..$ : chr "NAP"
  .. ..- attr(*, "term.labels")= chr "NAP"
  .. ..- attr(*, "order")= int 1
  .. ..- attr(*, "intercept")= int 1
```

Centering covariates

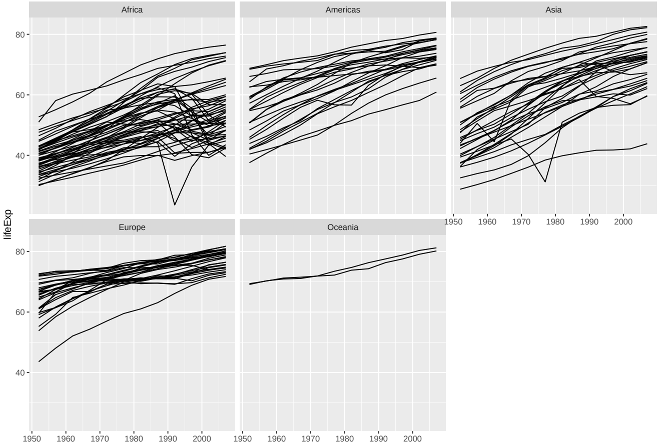
When dealing with numeric covariates, it often makes sense to center these variables before performing a regression.
- i.e. subtract the mean from each covariate.

This helps to make our parameters more meaningful.
- they now correspond to the average case in the data, rather than some extrapolated value.
- this is particularly useful for intercept parameters.

Fitting multiple models simultaneously

We may be interested in fitting many (many) models.
R has functionality to do this efficiently.

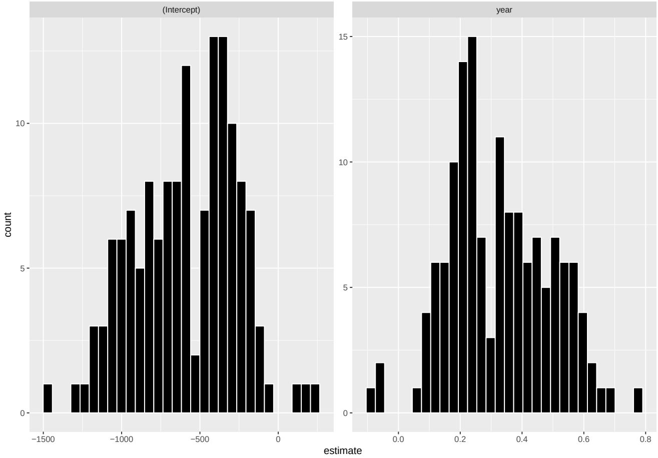
```
> library(broom)
> ggplot(gapminder, aes(x=year, y=lifeExp, group = country)) +
+   geom_line() +
+   facet_wrap(~continent)
```



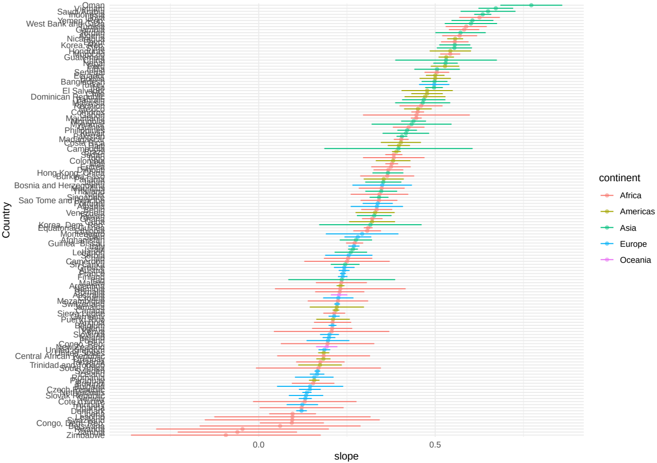
Fitting multiple models simultaneously

```
> gapminder_models <- gapminder |>
+   group_by(country) |>
+   nest() |>
+   mutate(model = map(data, ~lm(lifeExp~year, data = .x))) |>
+   mutate(coefs = map(model, tidy, conf.int = TRUE)) |>
+   unnest(coefs)
```

```
> ggplot(gapminder_models, aes(x = estimate, group = term)) +
+   geom_histogram(fill="black",col="white") +
+   facet_wrap(~term, scales= "free")
```



```
> slopes <- filter(gapminder_models, term == "year") %>%
+       arrange(desc(estimate))
> continents <- select(gapminder, country, continent) |>
+   distinct()
> slopes <- slopes |> left_join(continents)
Joining, by = "country"
> ggplot(slopes, aes(x = fct_reorder(country, estimate), y = est
+   #geom_histogram(fill="black",col="white") +
+   geom_point(alpha=0.5) +
+   geom_errorbar(aes(ymin=conf.low, ymax=conf.high), width=.05)
+   coord_flip() +
+   labs(y = "slope",
+        x = "Country") +
+   theme_minimal()
```


Oman
Vietnam
Saudi Arabia
Indonesia
Libya
Yemen, Rep.
West Bank and Gaza
Tunisia
Gambia
Jordan
Algeria
Nicaragua
Egypt
Korea, Rep.
Syria
Honduras
Morocco
Guatemala
China
Nepal
Peru
India
Senegal
Ecuador
Bolivia
Bangladesh
Turkey
Iran
El Salvador
Chile
Dominican Republic
Bahrain
Malaysia
Reunion
Mexico
Comoros
Gabon
Mauritania
Mongolia
Myanmar
Guinea
Philippines
Kuwait
Pakistan
Madagascar
Costa Rica
Haiti
Cambodia
Brazil
Sudan
Togo
Colombia
Mali
Eritrea
Djibouti
Hong Kong, China
Burkina Faso
Panama
Japan
Bosnia and Herzegovina
Mauritius
Thailand
Niger
Singapore
Sao Tome and Principe
Portugal
Albania
Benin
Venezuela
Taiwan
Ghana
Cuba
Korea, Dem. Rep.
Equatorial Guinea
Ethiopia
Montenegro
Spain
Afghanistan
Guinea-Bissau
Israel
Lebanon
Serbia
Cameroon
Sri Lanka
Greece
Austria
France
Finland
Iraq
Malawi
Argentina
Namibia
Somalia
Australia
Mozambique
Switzerland
Jamaica
Canada
Sierra Leone
Germany
Puerto Rico
Angola
Belgium
Nigeria
Kenya
Slovenia
Ireland
Poland
Congo, Rep.
New Zealand
United Kingdom
United States
Central African Republic
Uruguay
Tanzania
Trinidad and Tobago
South Africa
Sweden
Iceland
Romania
Paraguay
Burundi
Bulgaria
Czech Republic
Netherlands
Slovak Republic
Norway
Cote d'Ivoire
Hungary
Uganda
Denmark
Liberia
Lesotho
Swaziland
Congo, Dem. Rep.
Botswana
Rwanda
Zambia
Zimbabwe
Country
0.0
0.5
slope
continent
Africa
Americas
Asia
Europe
Oceania

Lab exercise 3/3

1. Fit a linear regression using 2007 gapminder data of the form `lm(gdpPercap ~ continent`, where gdpPercap is the new outcome variable `y`. Get information about the best-fitting line from the regression table. How do the regression results match up with those of an analysis of life expectancy by continent?
2. Extract the model coefficients and their 95 percent confidence intervals.
3. Plot the residuals vs the fitted values and comment on their distribution and patterns.
4. Identify the five countries with the five most negative residuals? What do these negative residuals say about their life expectancy relative to their continents life expectancy?
5. Repeat this process, but identify the five countries with the five most positive residuals. What do these positive residuals say about their life expectancy relative to their continents life expectancy?