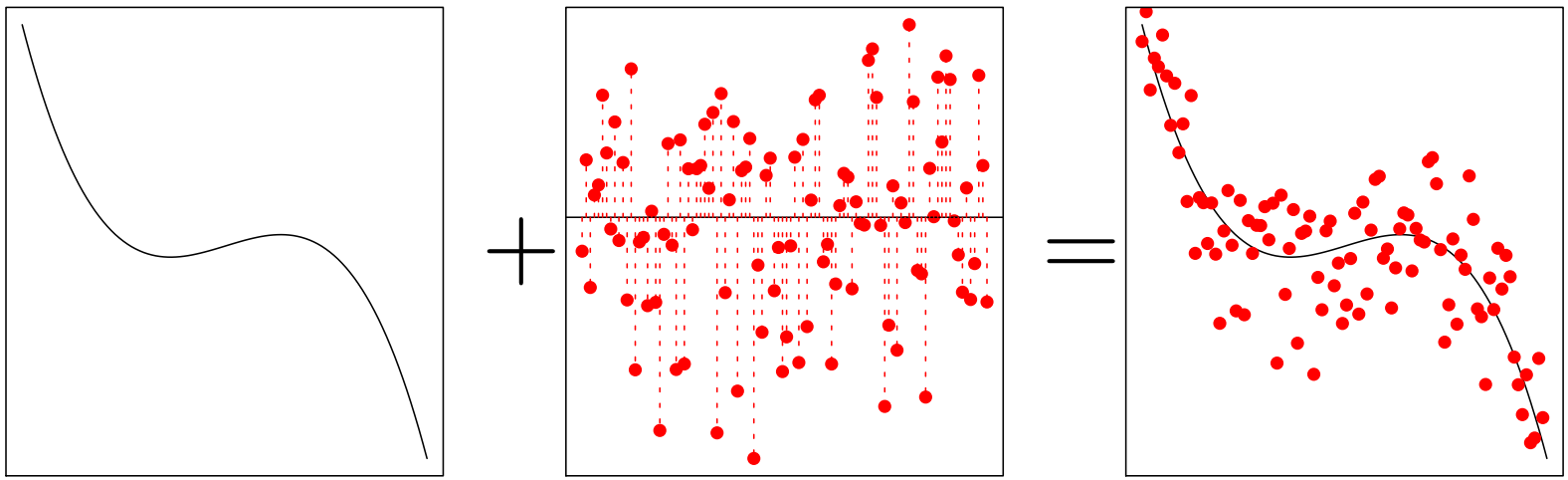
# Linear Models

September 12, 2014

# What is the point of a statistical model?

- (Underlying Pattern) + (Noise) = Data



- Pattern = hypothesized biological process
- Noise = random process and measurement error

Goal: Infer process by accounting for noise

# Dr. Box tells it like it is

*All models are wrong, but some are useful.*
– George E. P. Box

# What is a linear model?

A **linear model** is type of statistical model of the form:

- **Stochastic** / Noise component:

$$y_i \sim \text{Distribution(Parameters)}$$

- **Deterministic** / Process component:

$$\mathbb{E}[y_i] = \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_{ij} x_{ij}$$

For the $i$th observation: $y$ is response, $x$ are covariates, $\beta$ are regression coefficients. **Normal linear model**:

$$y \sim \mathcal{N}(\mu, \sigma)$$

$\mu$ and $\sigma$ are location and scale parameters of the normal distribution.

# Normal linear model

We can write this as **random variable** and **expected value**:

$$y_i \sim \mathcal{N}(\mu_i, \sigma), \quad \mathbb{E}[y_i] = \mu_i = \beta_1 x_{i1} + \beta_1 x_{i2} + ...$$

Or equivilantly as linear model plus **random errors**:

$$y_i = \beta_1 x_{i2} + \beta_2 x_{i2} + ... + \epsilon_i, \quad \epsilon \sim \mathcal{N}(0, \sigma)$$

Or equivalently as random variable where $\mu$ is linear model:

$$y_i \sim \mathcal{N}(\mu_i = [\beta_1 x_{i1} + \beta_1 x_{i2} + ...], \ \sigma)$$

For any technical description (not this one) a familiarity with matrix notation is mandatory!

# 'Deterministic' component

The linear model described above can incorporate:

- **Continuous** covariates
- **Categorical** covariates (as dummy variables)
- **Interactions** between covariates

Includes as special cases:

- Simple and multiple regression
- Analysis of variance (ANOVA)
- Analysis of covariance (ANCOVA)

In R we specify structure of linear model through the 'formula' syntax.

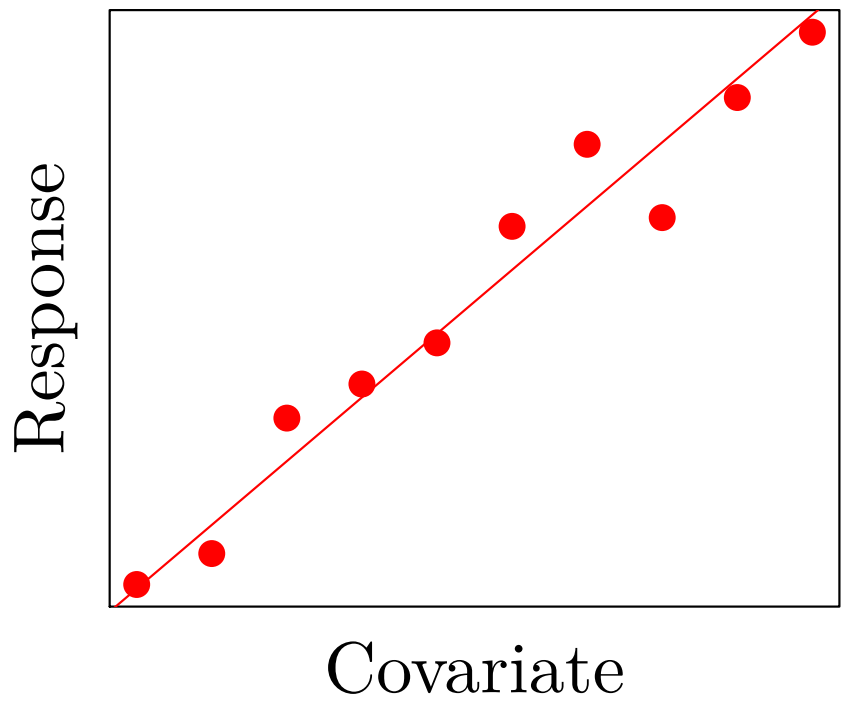
**response** $\sim$ **covariate** $+$ **factor** $+$ **...**

# 'Deterministic' component

Continuous covariates: just like in the equation for a line:

$$\mathbb{E}[y_i] = \alpha + \beta_1 x_i$$

$\alpha$ is **intercept**, $\beta$ is **slope**.




R syntax: **response** $\sim$ **1** $+$ **covariate** or **response** $\sim$ **covariate**

# 'Deterministic' component

Categorical covariates: each category is coded as a **dummy variable** $d_{ij}$.

- $d_{ij} = 1$ if $i$th observations belongs to category $j$.
- $d_{ij} = 0$ otherwise.

Think of this as a toggle (associated $\beta$ is turned on or off)

Can write linear model in terms of category means:

$$\mathbb{E}[y_i] = \beta_1 d_{i1} + \beta_2 d_{i2} + \beta_3 d_{i3} + ...$$

The $\beta_j$ are the mean value of $y$ for each category.

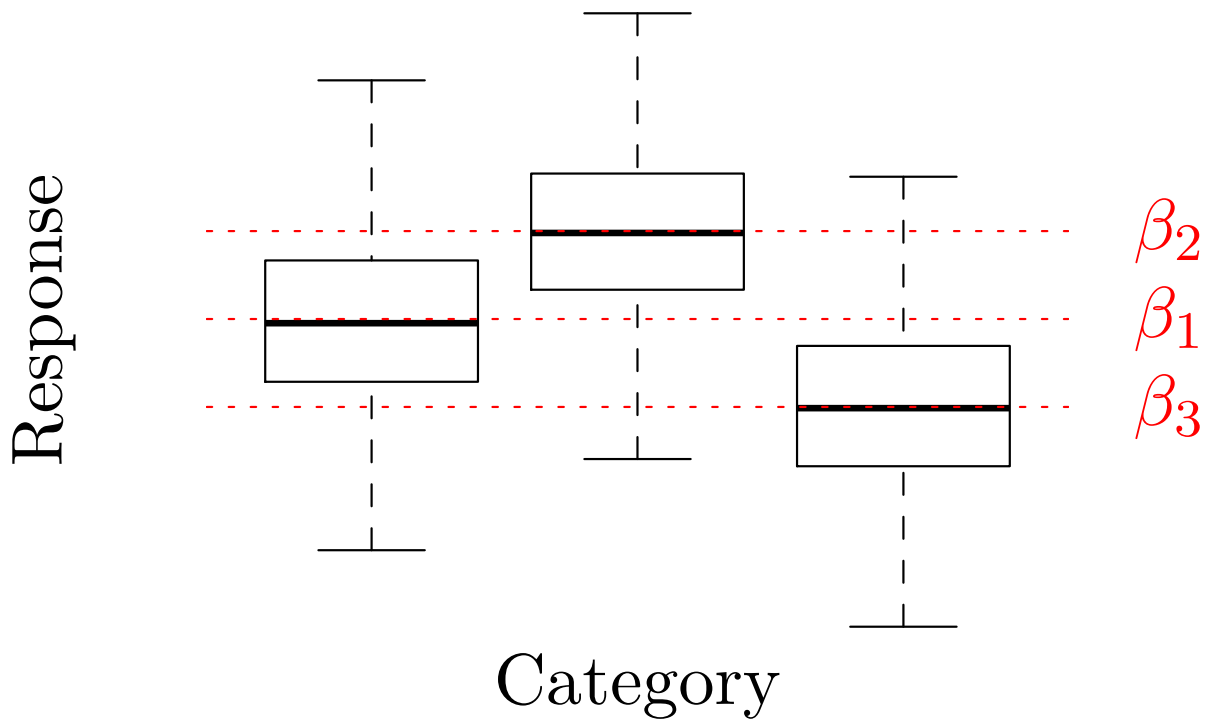
R syntax: **response ~ 0 + category**

# 'Deterministic' component

**Example:** for a three-category covariate, if the response belongs to category 2:

$$\mathbb{E}[y_i] = \beta_1 \cdot 0 + \beta_2 \cdot 1 + \beta_3 \cdot 0 = \beta_2 \cdot 1 = \beta_2$$

= the mean value of $y$ for category 2.



# 'Deterministic' component

Often we are interested in **mean differences** among groups. Set one of the categories as a **reference**:

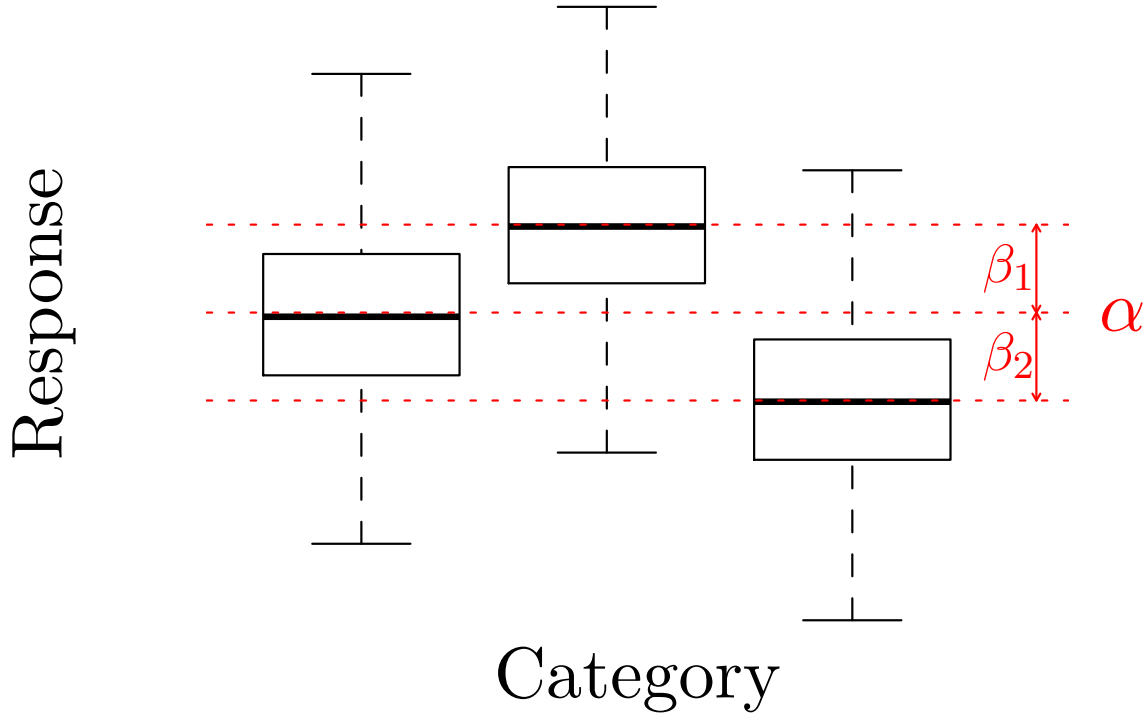
$$\mathbb{E}[y_i] = \alpha + \beta_2 d_{i2} + \beta_3 d_{i3} + ...$$

- $\alpha$ is the intercept = mean value of the response for the **reference**.
- $\beta_2$ and $\beta_3$ = **mean difference** between reference and categories 2 and 3.

This gives the exact same result as the previous example.

Ways of coding categorical variables to get the same outcome, with different coefficients, are called **contrasts**.

# 'Deterministic' component




With our three category example, if response belongs to category 2:

$$\mathbb{E}[y_i] = \alpha + \beta_2 \cdot 1 + \beta_3 \cdot 0 = \alpha + \beta_2$$

= mean value of category 2

R syntax: **response** $\sim$ **1 + category** OR **response** $\sim$ **category**

# 'Deterministic' component

With **multiple types of categories** (two-way ANOVA), define a reference level for both types of categories.

For example: **'red' vs. 'blue'** (first category type), and **'straight' vs. 'crooked'** (second category type)

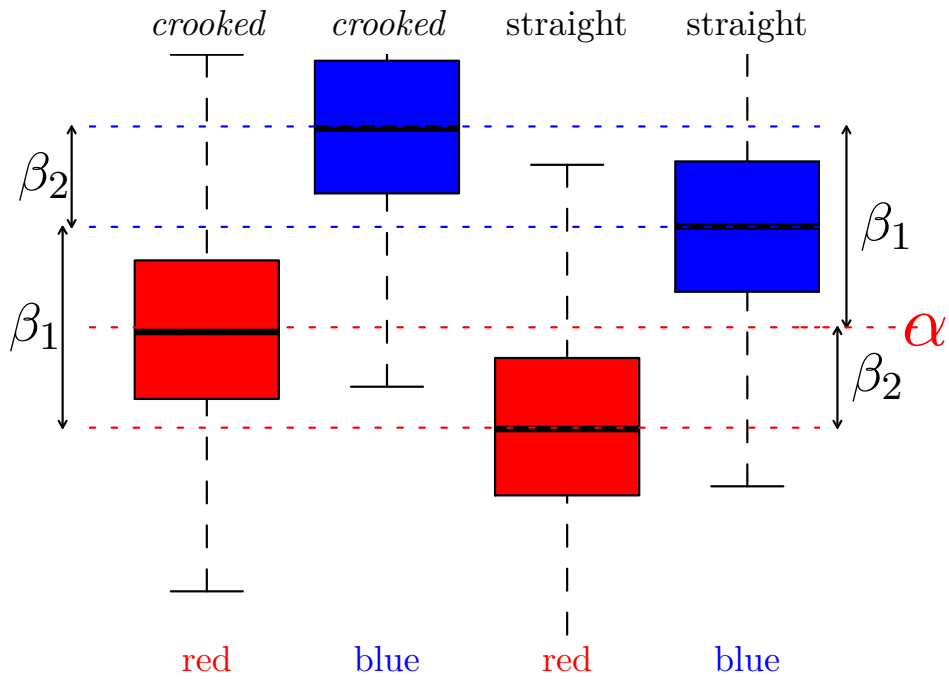
- 'red' and 'crooked' as reference combination
- $d_{i1} = 1$ if "blue", $d_{i2} = 1$ if "straight"

Model:

$$\mathbb{E}[y_i] = \alpha + \beta_1 d_{i1} + \beta_2 d_{i2}$$

$\alpha$ = mean response for 'red' and 'crooked'
$\beta_1$ = diff. in response between 'red' and 'blue'
$\beta_2$ = diff. in response between 'crooked' and 'straight'

# 'Deterministic' component




Given $\alpha, \beta_1, \beta_2$ calculate **four possible values** of response:

$\alpha + \beta_1 \cdot 0 + \beta_2 \cdot 0 = \alpha =$ mean value for **'crooked red'**

$\alpha + \beta_1 \cdot 1 + \beta_2 \cdot 0 = \alpha + \beta_1 =$ mean value for **'crooked blue'**

$\alpha + \beta_1 \cdot 0 + \beta_2 \cdot 1 = \alpha + \beta_2 =$ mean value for **'straight red'**

$\alpha + \beta_1 \cdot 1 + \beta_2 \cdot 1 = \alpha + \beta_1 + \beta_2 =$ mean value for **'straight blue'**

# 'Deterministic' component

Continuous and categorical covariates can be combined:

$$\mathbb{E}[y_i] = \alpha + \beta_2 d_{i2} + \beta_3 x_{i3}$$

In this case the dummy variable and associated coefficient, **combine into the intercept**. i.e. where $d_{i2} = 1$:

$$\begin{aligned}\mathbb{E}[y_i] &= \alpha + \beta_2 \cdot 1 + \beta_3 x_{i3} \\ &= (\alpha + \beta_2) + \beta_3 x_{i3}\end{aligned}$$

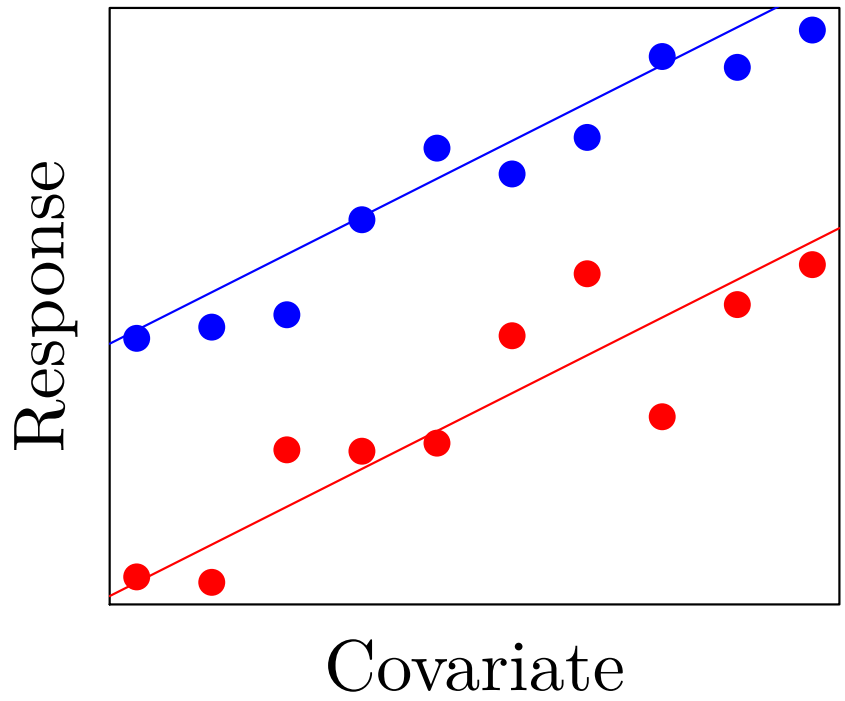
$\alpha$ is the **intercept** for category 1

$\beta_2$ is the **mean difference** between the intercept for category 2 and the intercept for category 1.

$\beta_3$ is the **slope** associated with continuous covariate, $x_{.3}$.

# 'Deterministic' component

This linear model (classic ANCOVA) codes for **two** lines, with the **same slope** but **different intercepts**.




R syntax: **response** $\sim$ **covariate** $+$ **category**

# 'Deterministic' component

Covariates can interact. The interaction term takes the form:

(regression coefficient) · (covariate one) · (covariate two)

For categorical by continuous interaction, the **interaction combines with a slope**:

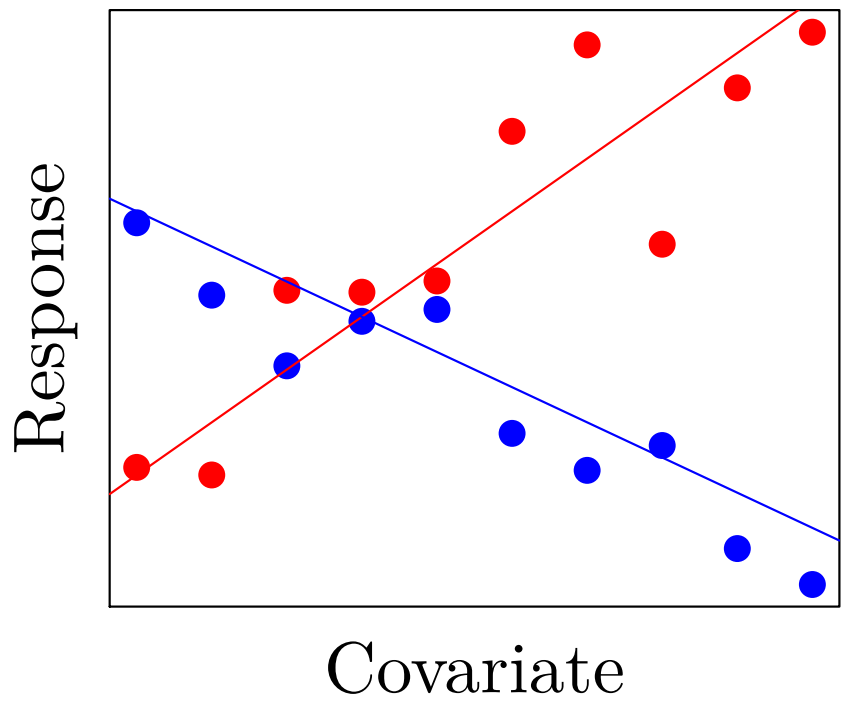
$$\begin{aligned}\mathbb{E}[y_i] &= \alpha + \beta_2 d_{i1} + \beta_3 x_{i2} + \beta_4 d_{i1} x_{i2} \\ &= \alpha + \beta_2 \cdot 1 + \beta_3 x_i + \beta_4 \cdot 1 \cdot x_{i2} \\ &= (\alpha + \beta_2) + (\beta_3 + \beta_4) x_{i2}\end{aligned}$$

For categorical by categorical interaction, the **interaction combines with an intercept**:

$$\begin{aligned}\mathbb{E}[y_i] &= \alpha + \beta_2 d_{i1} + \beta_3 d_{i2} + \beta_4 d_{i1} d_{i2} + \beta_5 x_i \\ &= \alpha + \beta_2 \cdot 1 + \beta_3 \cdot 1 + \beta_4 \cdot 1 \cdot 1 + \beta_5 x_i \\ &= (\alpha + \beta_2 + \beta_3 + \beta_4) + \beta_5 x_i\end{aligned}$$

# 'Deterministic' component

For categorical by continuous interaction, the **regression coefficient for the interaction** is the **difference in slope** between a category and the reference category.



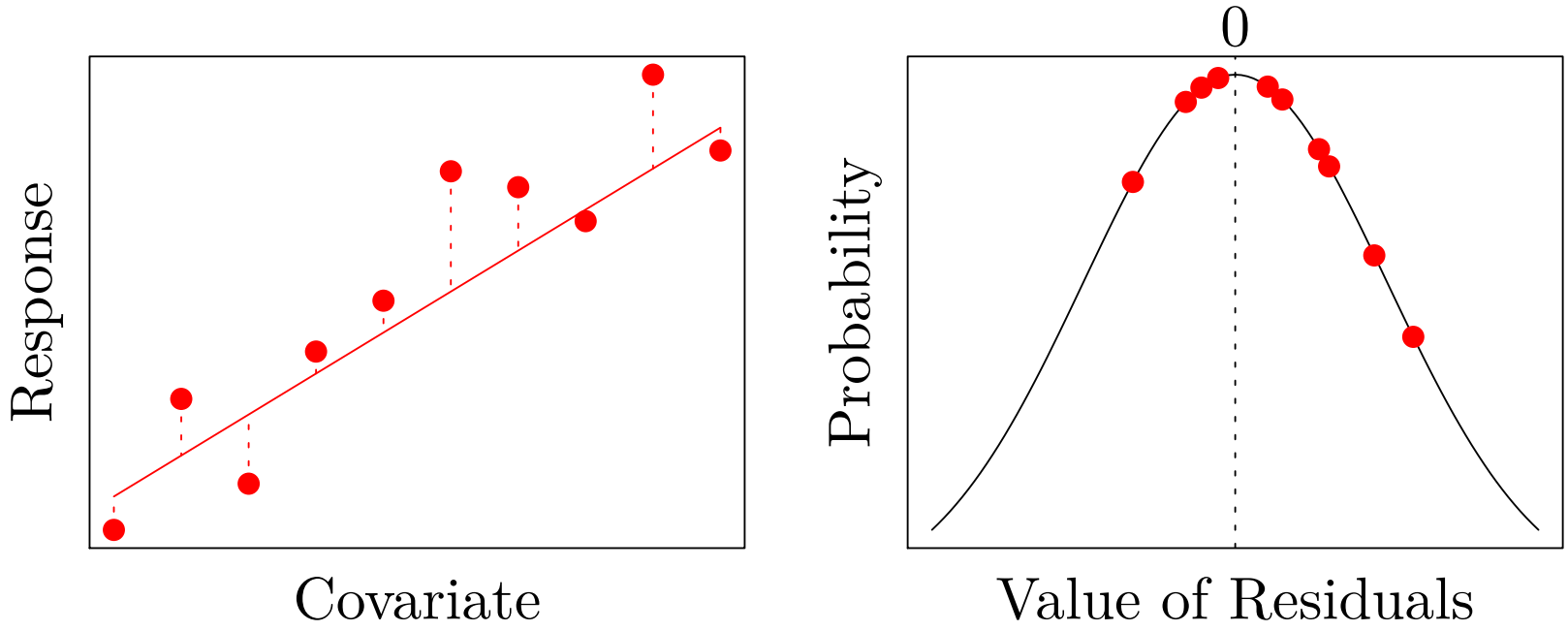

This corresponds to a classical ANCOVA with interaction.
R syntax: **response** $\sim$ **covariate** + **category** + **covariate:category** OR

**response** $\sim$ **covariate** * **category**

# 'Stochastic' component

Deterministic component gives the **fitted** values.

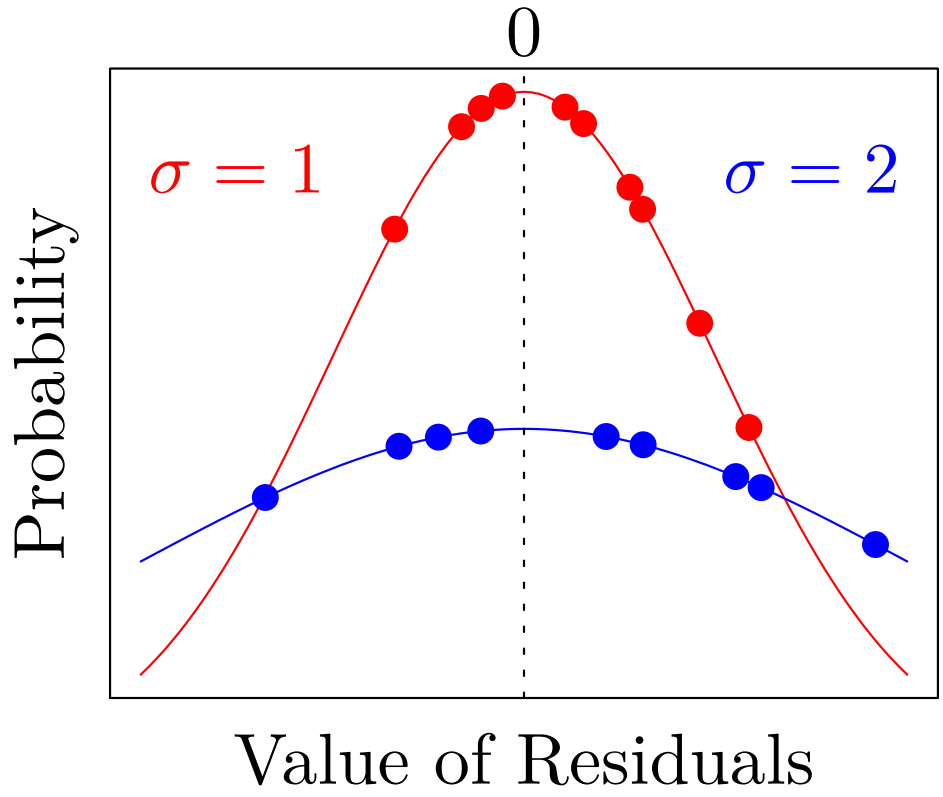
The difference between the observed data and the fitted values are the **residuals**.




For a normal linear model, we use a **normal probability distribution** with $\mu = 0$, to describe the random properties of the residuals.

# 'Stochastic' component

The scale parameter (aka standard deviation aka $\sigma$) determines the spread of the residuals. In the plot below, the red and blue points have the same **quantiles**, but are from distributions with different scales.




A model with a single scale parameter – where the residuals all have the same scale regardless of fitted value – is a **homoskedastic** model.

# Degrees of freedom

**Degrees of freedom** can be thought of as the number of independent bits of information we have to estimate a given measure.

A normal linear model has **residual degrees freedom** $N - p$ where $N$ is number of observations and $p$ is effective number parameters.

A model is said to be **overfit** when the deterministic portion is describing random noise instead of an underlying pattern.

# Likelihood

We have specified the deterministic and stochastic components of our model(s).

We have collected data that corresponds to the model(s).

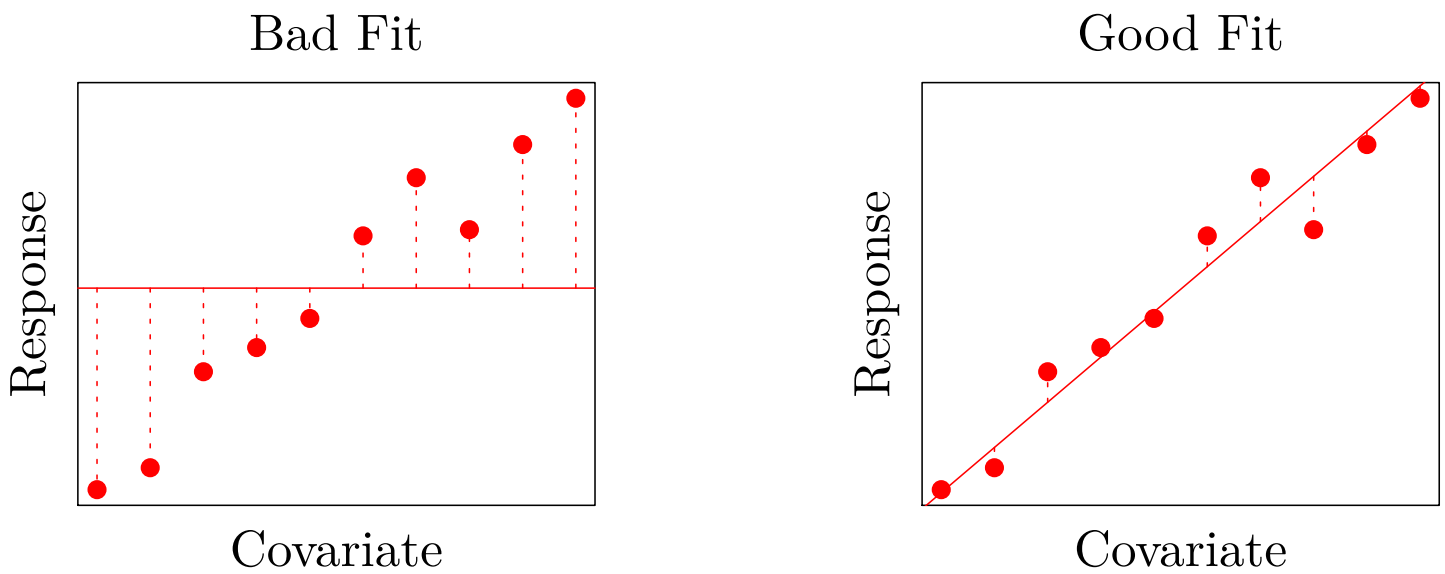
Now we need to **fit** the model(s) – estimate the parameters ($\alpha$, $\beta$, etc.) – and to **test hypotheses** about the model(s).

**Likelihood** provides a framework for doing this.

# Likelihood

Negative **likelihood** is one of many **loss functions**.

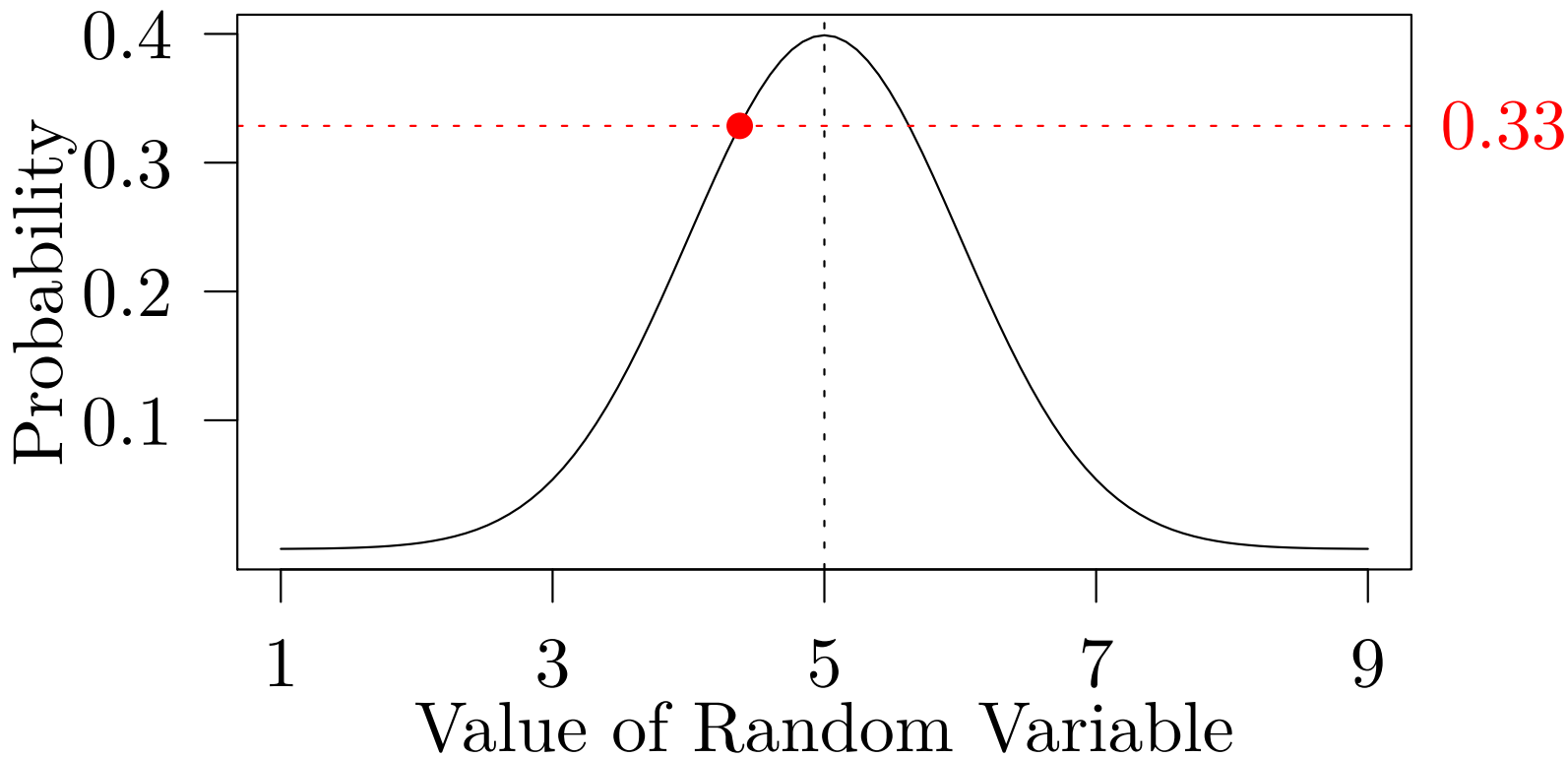
A loss function is a measure of how poorly a model fits the data. A familiar example is the sum of squared residuals (the *least squares* criterion):



# Likelihood

For a given probability distribution with specified parameters, we have a **probability density function** (pdf), which gives the probability of a given value of a random variable.

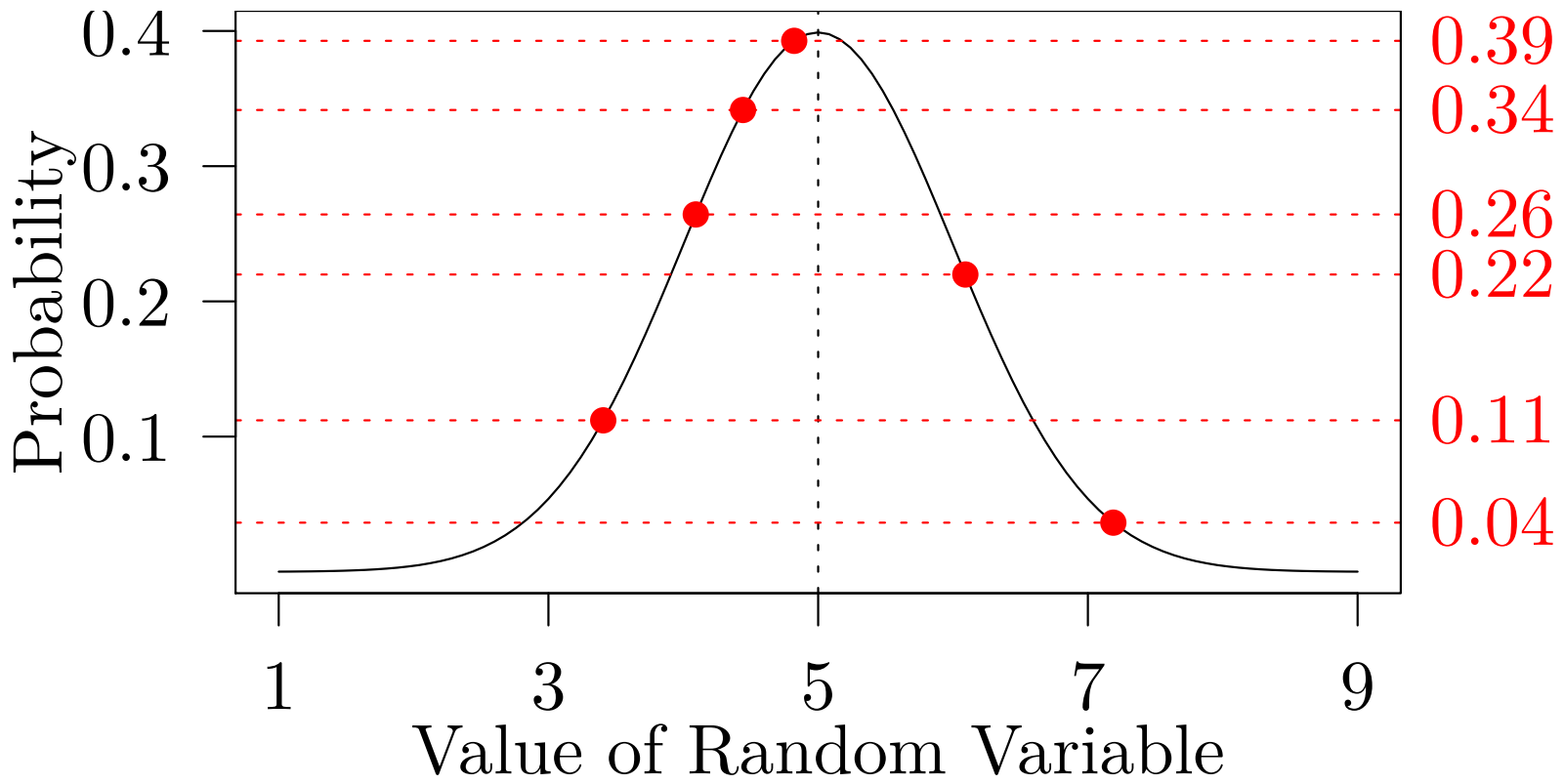
For example, for a single data point from normal distribution with $\mu = 5$:



# Likelihood

What is the probability of **multiple observations**?

Recall from basic probability: the probability of $n$ independent events is the **product of the probabilities** of each event.



# Likelihood

The product of these probabilities is:

$$0.04 \cdot 0.11 \cdot 0.22 \cdot 0.26 \cdot 0.34 \cdot 0.39 = 3.3 \cdot 10^{-5}$$

This is the **likelihood** of $\mu = 5$, **given these data** and the scale parameter (fixed at 1 for this example):

$$\mathcal{L}(\mu = 5 | y, \sigma)$$

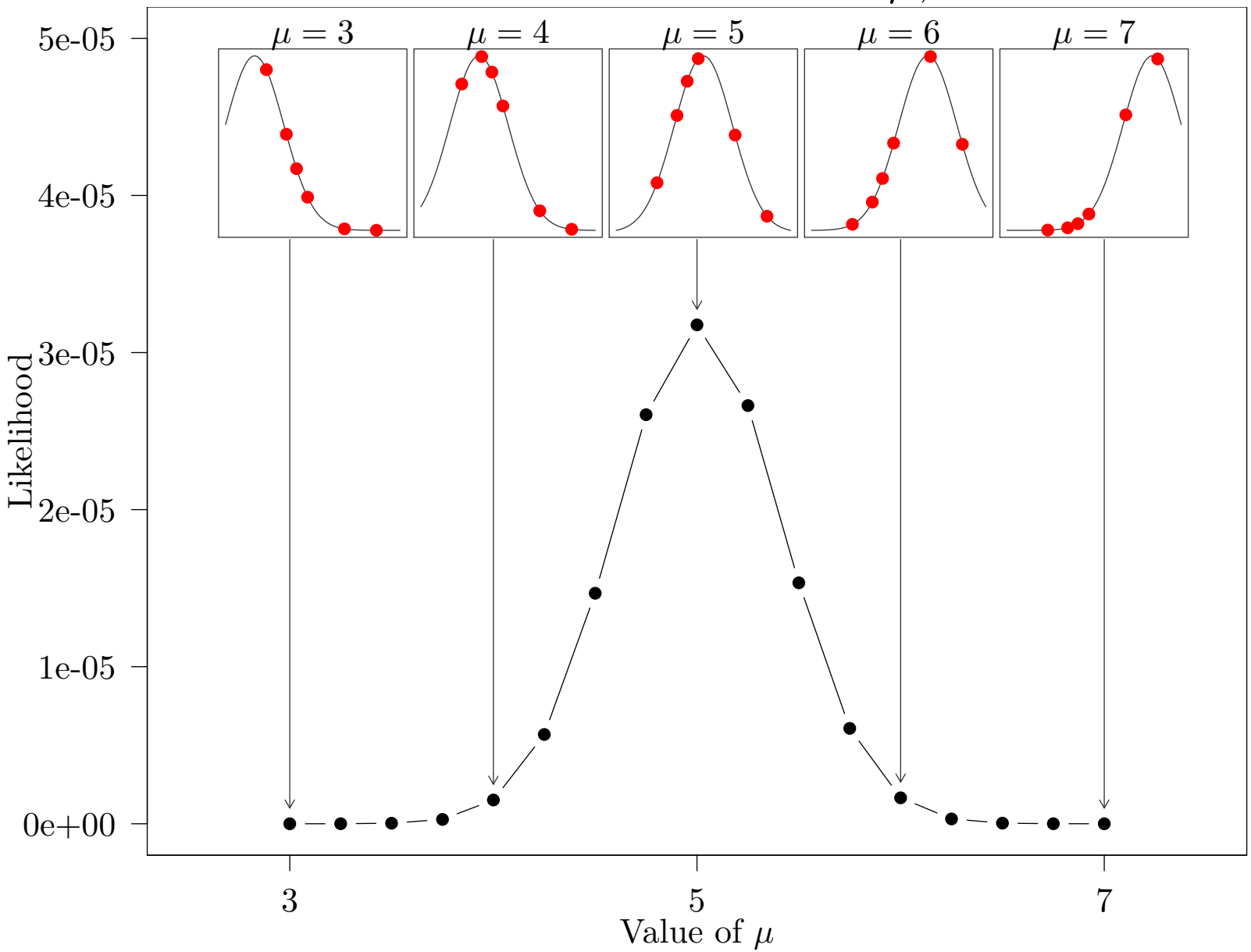
Because the product of many probabilities is very small, we usually use the **log-likelihood**.

$$\log \mathcal{L}(\mu = 5 | y, \sigma)$$

For clarity we'll stick with the (unlogged) likelihood in this example.

# Likelihood

What is likelihood at **different values of** $\mu$, for the same data?



# Fun facts about likelihood

This is the **likelihood surface** for $\mu$.

The peak of this surface is the **maximum likelihood estimate** (MLE).

The **curvature of the liklihood surface** relates to the precision of the MLE. If wide and flat, many values have similar likelihood. If narrow and steep, few values have similar likelihood.

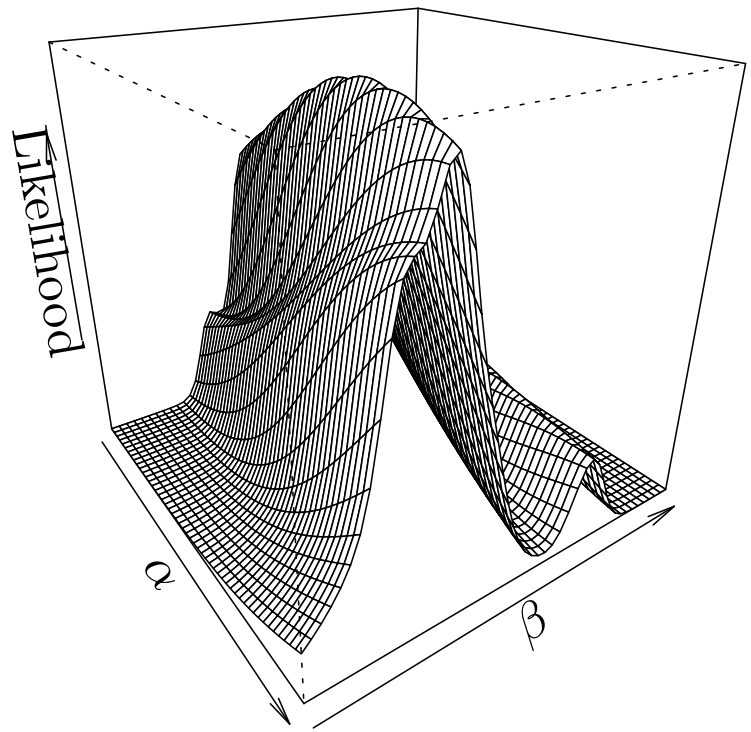
The data are random, and the MLE is a function of the data: thus the **MLE is random**.

The MLE is asymptotically normally distributed, for a normal model.

The **standard error** (SE) of the MLE, is an estimate of standard deviation of the distribution of the MLE.

# Likelihood

For multiple parameters (i.e. $\alpha$ and $\beta$ in a simple regression), the likelihood surface is two-dimensional. For $y = \alpha + \beta x$, where $x = [1 : 10]$:




Then the sampling distribution of the MLE is **multivariate normal**, with a variance-covariance matrix called the **inverse information matrix**.

# Likelihood

Depending on the statistical model, the MLE can be found through **analytical methods, optimization,** etc.

For linear models, $\sigma$ is often treated as a nuisance parameter and is estimated with the unbiased estimator

$$\sigma^2 = \frac{1}{N-p}\sum_{i=0}^{N}(y_i - \mathbb{E}[y_i])^2$$

For the normal linear model, **ordinary least squares** finds the MLE.

# Workflow for modelling

1. Generate biological hypotheses
2. Develop models for hypotheses
3. Gather data to test models
4. Fit models to data
5. Model selection
6. Diagnose best model(s)
7. Test hypotheses with models

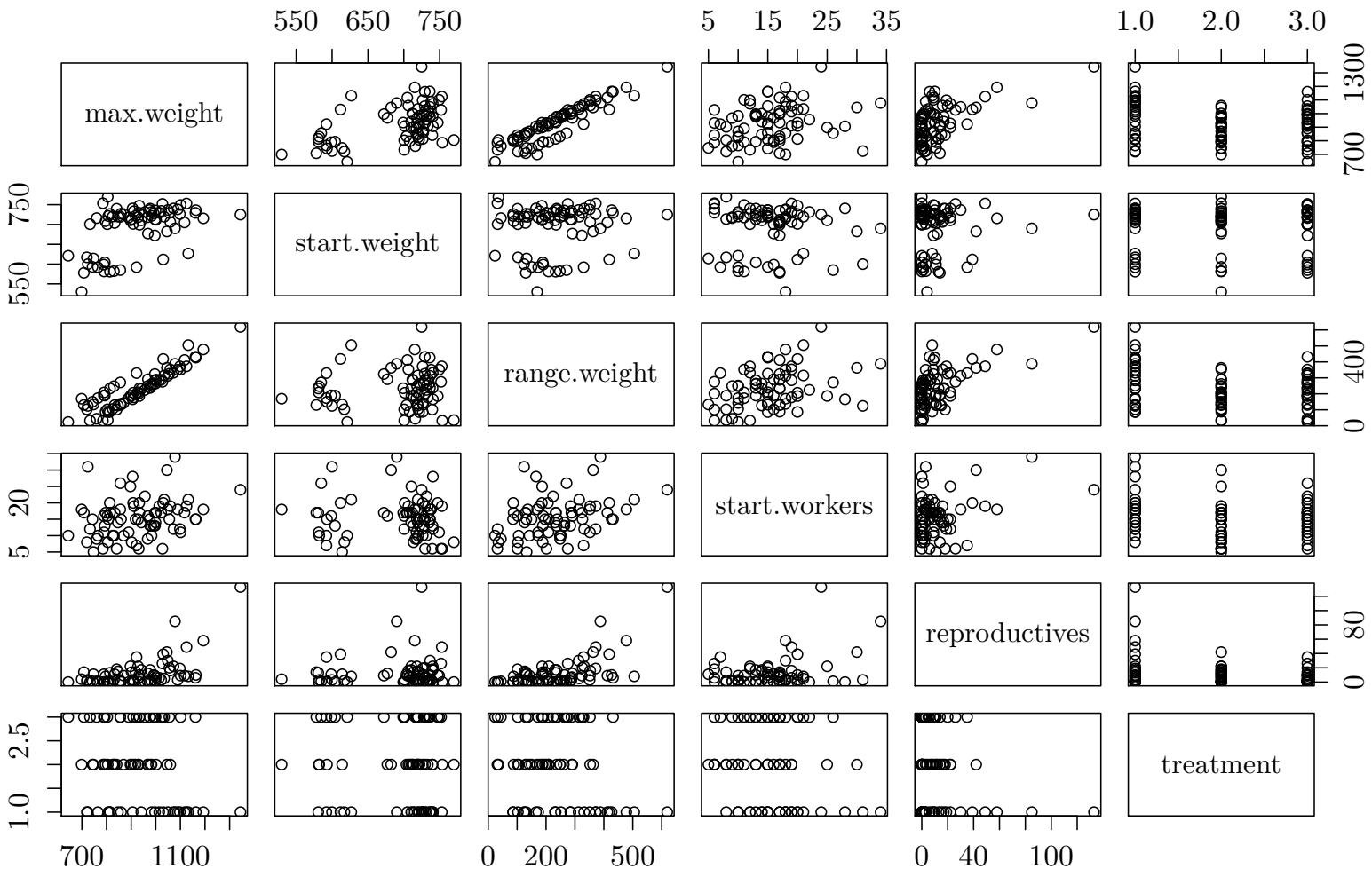
# Motivating example

From Whitehorn et al. Science 2012: *Bombus* exposed to neonicatinoid pesticides. Do neonicatinoids impact colony growth?

- Response : the maximum weight a colony reaches, **max.weight** in data
- Example of a non-normal response : number of queens produced, **reproductives** in data
- Predictor of interest : neonicatinoid treatment (CONTROL, LOW, HIGH), **treatment** in data
- Covariates : weight and workers by colony at start, **start.weight, start.workers** in data

# Motivating example

```
Bombus = read.csv("Bombus_pesticide.csv")
pairs(Bombus)
```



# Fitting a linear model

**lm(<formula>, <data>)** is the workhorse for the normal linear model.

Confusing terminology:

- **'General linear model'** is sometimes used for the normal linear model
- **'Generalized linear model'** refers to linear models of exponential-family distributions, of which normal is one

R code for fitting model:

```
weight_lm <- lm(max.weight ~ start.weight +
    start.workers + treatment, data = Bombus)
```

# Output from a linear model

**print(<lm object>)** will return a brief summary.

```
print(weight_lm)

##
## Call:
## lm(formula = max.weight ~ start.weight + start.workers + treatment,
##     data = Bombus)
##
## Coefficients:
##   (Intercept)   start.weight  start.workers  treatmentHigh   treatmentLow
##         58.49           1.20           5.44         -77.39         -53.83
```

# Output from a linear model

**summary(<lm object>)** will return a longer summary.

```
summary(weight_lm)
```

```
##
## Call:
## lm(formula = max.weight ~ start.weight + start.workers + treatment,
##     data = Bombus)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -222.53  -70.08    4.79   72.56  288.54
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     58.489    166.710    0.35    0.727
## start.weight     1.195      0.226    5.30  1.3e-06 ***
## start.workers    5.445      2.273    2.40    0.019 *
## treatmentHigh  -77.386     32.837   -2.36    0.021 *
## treatmentLow   -53.831     31.975   -1.68    0.097 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 111 on 70 degrees of freedom
## Multiple R-squared:  0.385, Adjusted R-squared:  0.35
## F-statistic: 10.9 on 4 and 70 DF,  p-value: 5.96e-07
```

# Output from a linear model

**summary(<lm object>)$coef** will return just the coefficient table.

```
summary(weight_lm)$coef
```

```
##                Estimate Std. Error t value  Pr(>|t|)
## (Intercept)      58.489   166.7101  0.3508 7.268e-01
## start.weight      1.195     0.2256  5.2969 1.295e-06
## start.workers     5.445     2.2726  2.3958 1.926e-02
## treatmentHigh   -77.386    32.8372 -2.3567 2.124e-02
## treatmentLow    -53.831    31.9751 -1.6835 9.673e-02
```

# Output from a linear model

Many useful components are stored within model object.

```
str(weight_lm, max.level = 1, give.attr = F,
    width = 40, strict.width = "cut")

## List of 13
##  $ coefficients : Named num [1:5] 58.4..
##  $ residuals    : Named num [1:75] 15...
##  $ effects      : Named num [1:75] -80..
##  $ rank         : int 5
##  $ fitted.values: Named num [1:75] 981..
##  $ assign       : int [1:5] 0 1 2 3 3
##  $ qr           :List of 5
##  $ df.residual  : int 70
##  $ contrasts    :List of 1
##  $ xlevels      :List of 1
##  $ call         : language lm(formula ..
##  $ terms        :Classes 'terms', 'for..
##  $ model        :'data.frame': 75 obs...
```

# Output from a linear model

Extractor functions return raw and derived output.

**formula(<lm object>)** : returns formula
**resid(<lm object>)** : returns residuals
**fitted(<lm object>)** : returns fitted values
**coef(<lm object>)** : returns estimated coefficients
**vcov(<lm object>)** : variance-covariance of coefficients
**model.matrix(<lm object>)** : matrix of predictors
**logLik(<lm object>)** : returns log-likelihood of model
**summary(<lm object>)$sigma** : estimate of standard deviation
**df.residual(<lm object>)** : residuals degrees freedom

```
formula(weight_lm)

## max.weight ~ start.weight + start.workers + treatment
```

# Model selection

Often we have a model set representing **competing hypotheses**, and need to evaluate which model is the best.

Model selection can only occur between models with the same response data (the same observations).

Two approaches I discuss:

- via **information criteria**: for any models with the same response
- via **hypothesis testing** for nested models (will discuss later)

Both use likelihood and are based on the principle of parsimony.

# Model selection

**Information criteria** provide a summary value for each model.

All are based on the following:

$$(\text{Measure of Fit}) + (\text{Penalty for Complexity})$$

Most common for maximum-likelihood $\Longrightarrow$ **Akaike's Information Criterion** (AIC):

$$\text{AIC} = -2 \cdot (\text{LogLikelihood}) + 2 \cdot (\text{Number of Parameters})$$

Many variants exist: differ in the penalty for complexity.

# Model selection

The model with the lowest AIC is preferred.

- Models **within 2** AIC units from the best model are considered **nearly equivalent**.
- Models **between 2-10** AIC units from the best model are considered to have **weak support**.
- Models with **greater than 10** AIC above the best model are considered to have **no support**.

We typically summarize by $\Delta$**AIC**:

$$\Delta\text{AIC} = (\text{AIC of model}) - (\text{AIC of best model})$$

# Model selection

Example: using **max.weight** as response,

- Full model : $\sim$ start.weight + start.workers + treatment
- Reduced model : $\sim$ start.weight
- Null model : $\sim 1$

We use **update()** to remove terms from our full model.

```
full_model <- weight_lm
reduced_model <- update(full_model, . ~ start.weight)
null_model <- update(full_model, . ~ 1)
```

We use **ICtab()** in package **bbmle** to create an AIC table for all models:

```
bbmle::ICtab(full_model, reduced_model, null_model,
    type = "AIC", base = T, weights = T)
```

```
##                AIC   df dAIC  weight
## full_model    926.2 6    0.0 0.9924
## reduced_model 935.9 3    9.7 0.0076
## null_model    954.6 2   28.4 <0.001
```

# Model selection

**AIC(<lm object>)** extracts AIC from a given model object.

```
AIC(full_model)
```

```
## [1] 926.2
```

**AICc(<lm object>)** in package **bbmle** extracts second-order AIC (corrected for small sample size).

```
bbmle::AICc(full_model)
```

```
## [1] 927.4
```

**step()** performs automatic model selection with AIC, but is not a good idea (thoughtless model selection returns thoughtless model)

In R, models with **transformed** responses (log, etc.) cannot naively be compared to **untransformed** equivalent (more on this later)

# Diagnostics for linear models

Why **diagnostics** in addition to model selection? A model can be the best in a set of models, and still be rubbish.

Three themes:

- Check model assumptions
- Detect poorly fit data points (outliers)
- Detect influential data points

R has many capabilities for model diagnostics: only a few covered here.

Check out package **car** and associated book.

# Diagnostics for linear models

Residuals (in various forms) are the primary tool for diagnostics.

**Standardized residuals** are scaled by overall residual variance.

**Studentized residuals** are scaled by residual-specific variance.

```
resid(weight_lm)[1:5]  # raw
```

```
##       1       2       3       4       5
##   15.68 -148.69  123.50 -222.53 -190.04
```

```
rstandard(weight_lm)[1:5]  # standardized
```

```
##       1       2       3       4       5
##  0.1448 -1.3718  1.1353 -2.0521 -1.7995
```

```
rstudent(weight_lm)[1:5]  # studentized
```

```
##       1       2       3       4       5
##  0.1438 -1.3807  1.1377 -2.1016 -1.8294
```

# Diagnostics for linear models

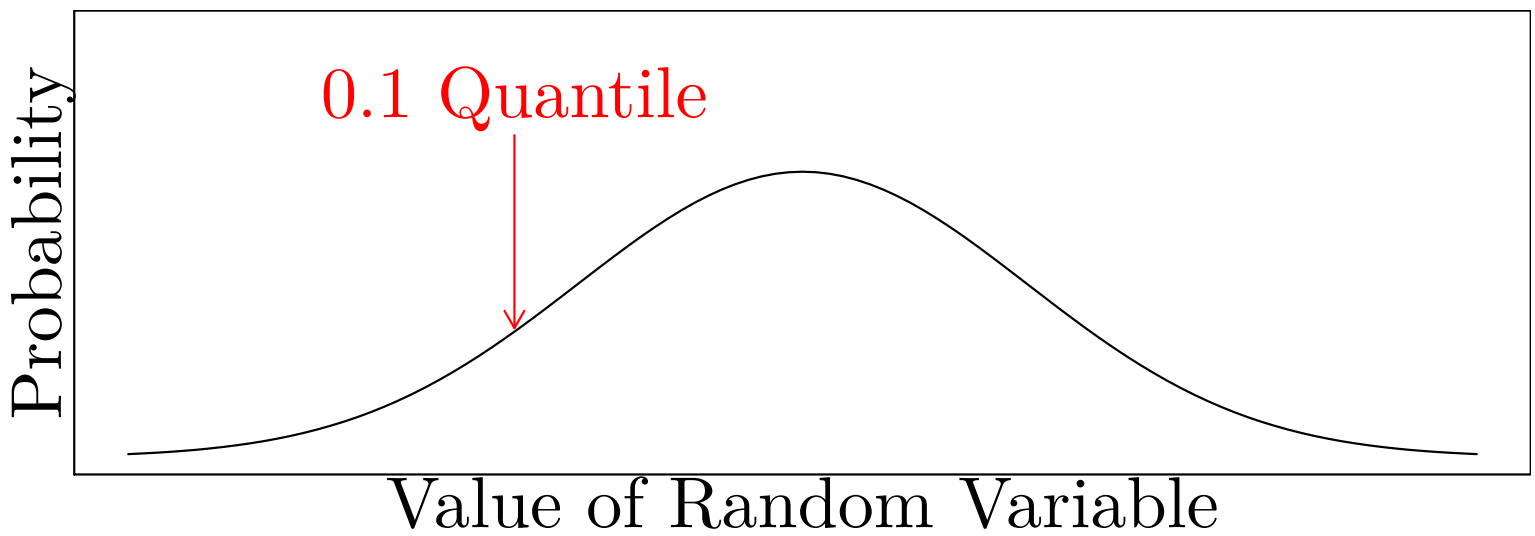
Assumption 1: **Normality of residuals**

Note that **residuals** need to be normal, not **data**

$\Longrightarrow$ obvious from this formula:

$$y_i = \beta_1 x_{i2} + \beta_2 x_{i2} + ... + \epsilon_i, \quad \epsilon \sim \mathcal{N}(0, \sigma)$$

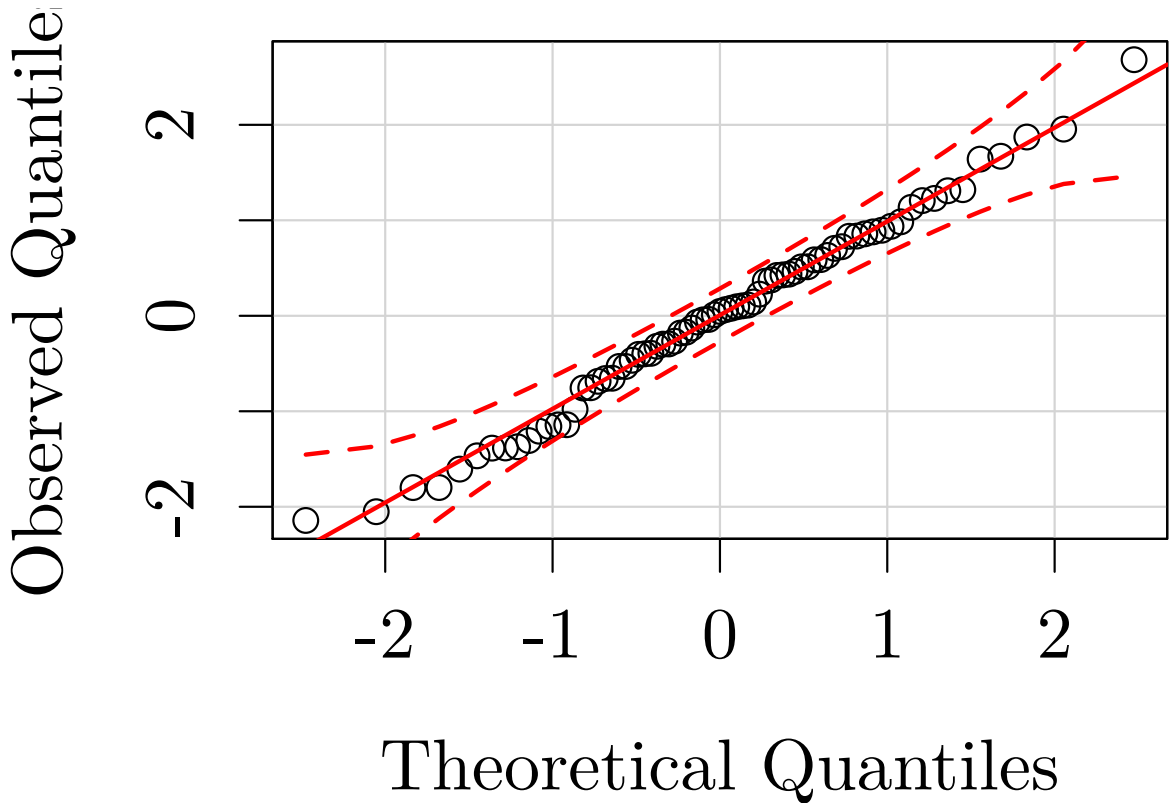
**Quantile**: the value of a random variable beneath which variates are drawn with a given probability.



# Diagnostics for linear models

Quantile-Quantile plot: observed quantiles vs. theoretical quantiles

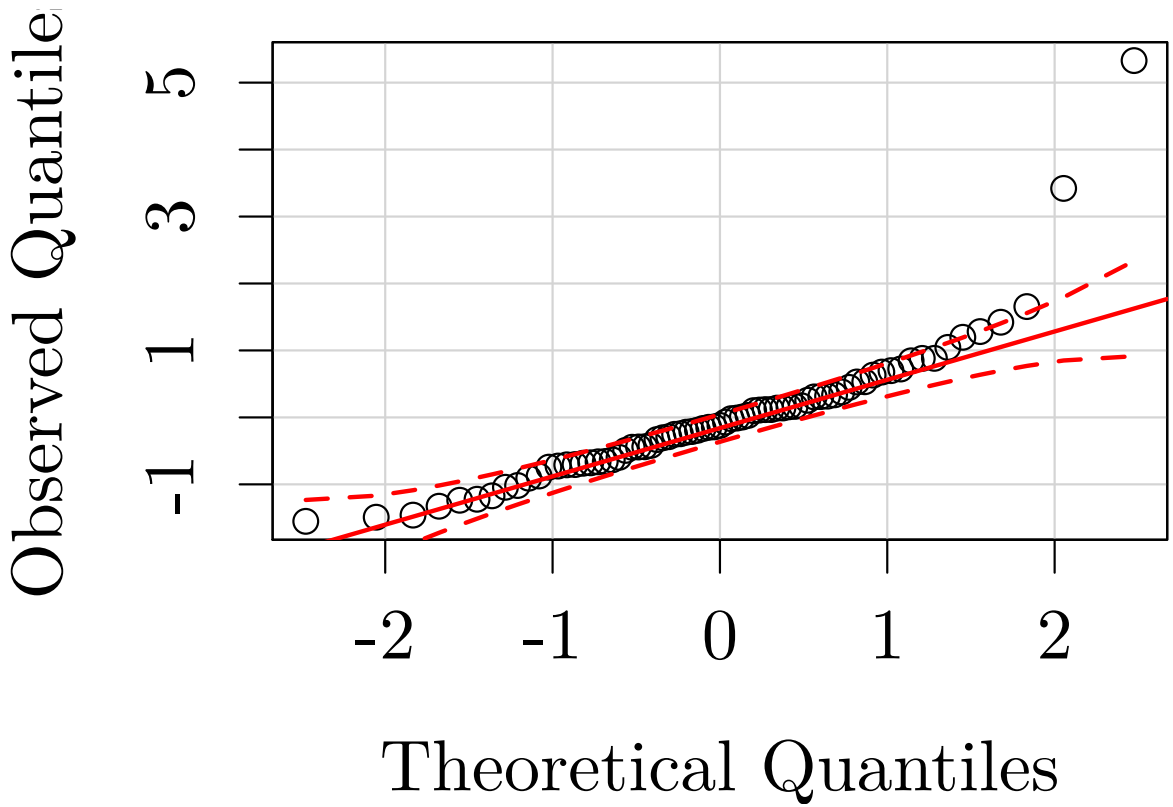
```
car::qqPlot(rstandard(weight_lm), xlab = "Theoretical Quantiles",
    ylab = "Observed Quantiles")
```



# Diagnostics for linear models

Example of non-normal residuals:

```
reprod_lm <- lm(reproductives ~ max.weight +
    treatment, data = Bombus)
car::qqPlot(rstandard(reprod_lm), xlab = "Theoretical Quantiles",
    ylab = "Observed Quantiles")
```




**What can you do?** use GLM, transform response

# Diagnostics for linear models

Assumption 2: **Homoskedasticity**

Constant variance of residuals. Again apparent from formula for linear model.

$$y_i = \beta_1 x_{i2} + \beta_2 x_{i2} + ... + \epsilon_i, \quad \epsilon \sim \mathcal{N}(0, \sigma)$$

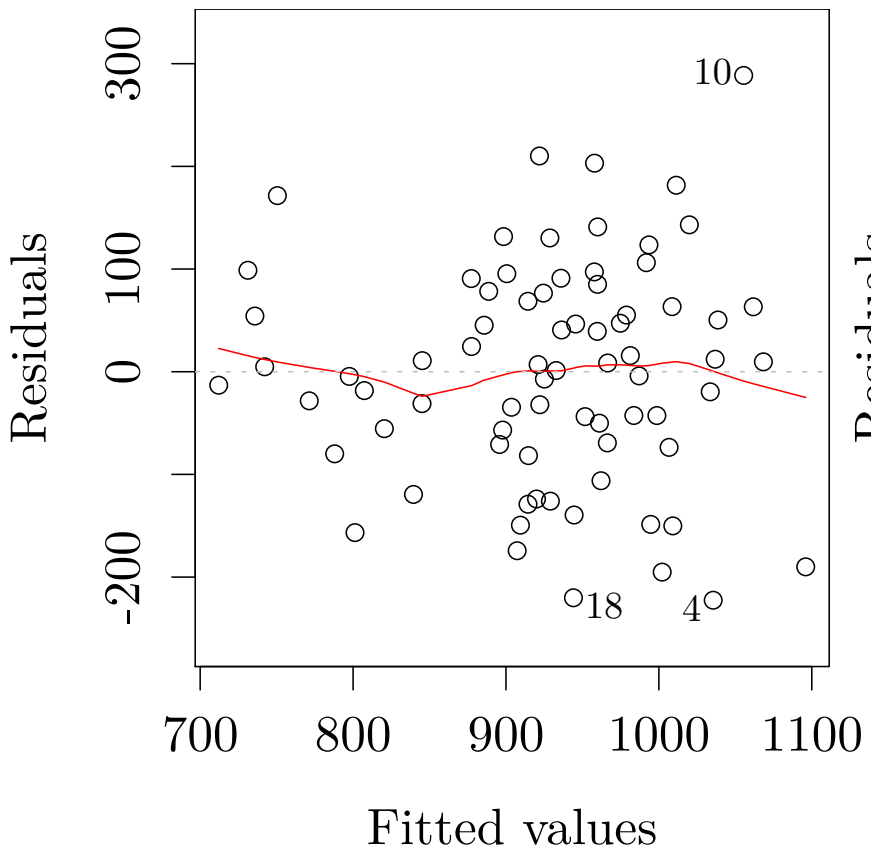
**One** scale variable for **all** residuals regardless of fitted value.

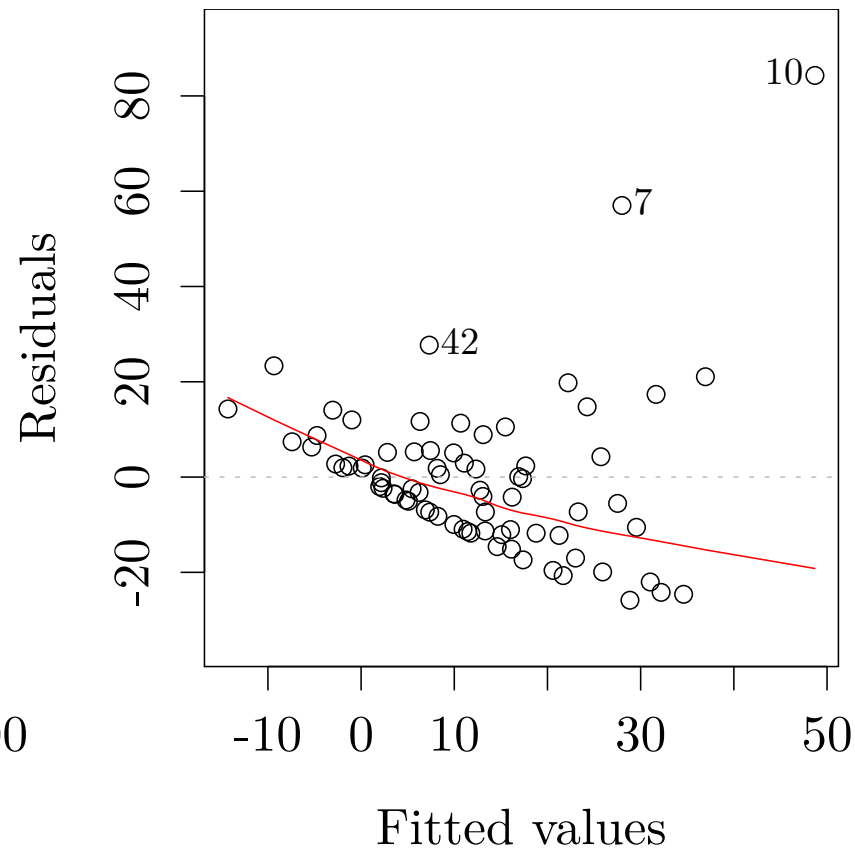
If **heteroskedastic**, can use **gls()** to fit model with non-constant variance.

# Diagnostics for linear models

Check by plotting residuals against fitted values.

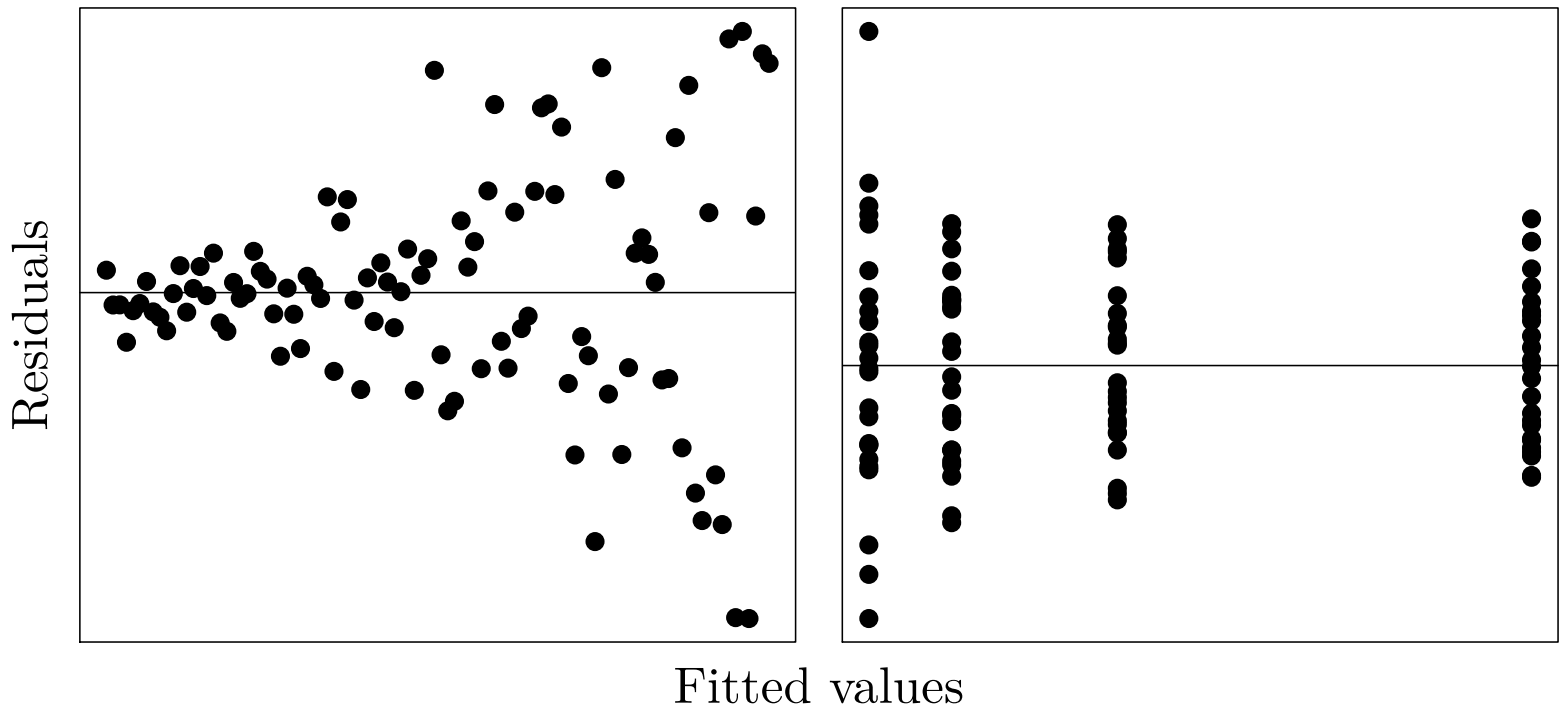
```
plot(weight_lm, which = 1)   # homoskedastic
plot(reprod_lm, which = 1)   # heteroskedastic
```



# Diagnostics for linear models

Some common patterns of heteroskedasticity:



# Diagnostics for linear models

Assumption 3: **Linearity**

If underlying process is nonlinear, a linear model will not be adequate.

**CERES** plot: plots partial residuals for $j$th covariate against $j$th covariate.

**Partial Residuals**: for $j$th covariate, residuals from the fitted value with $j$th covariate dropped. I.e. for 3rd covariate in model:

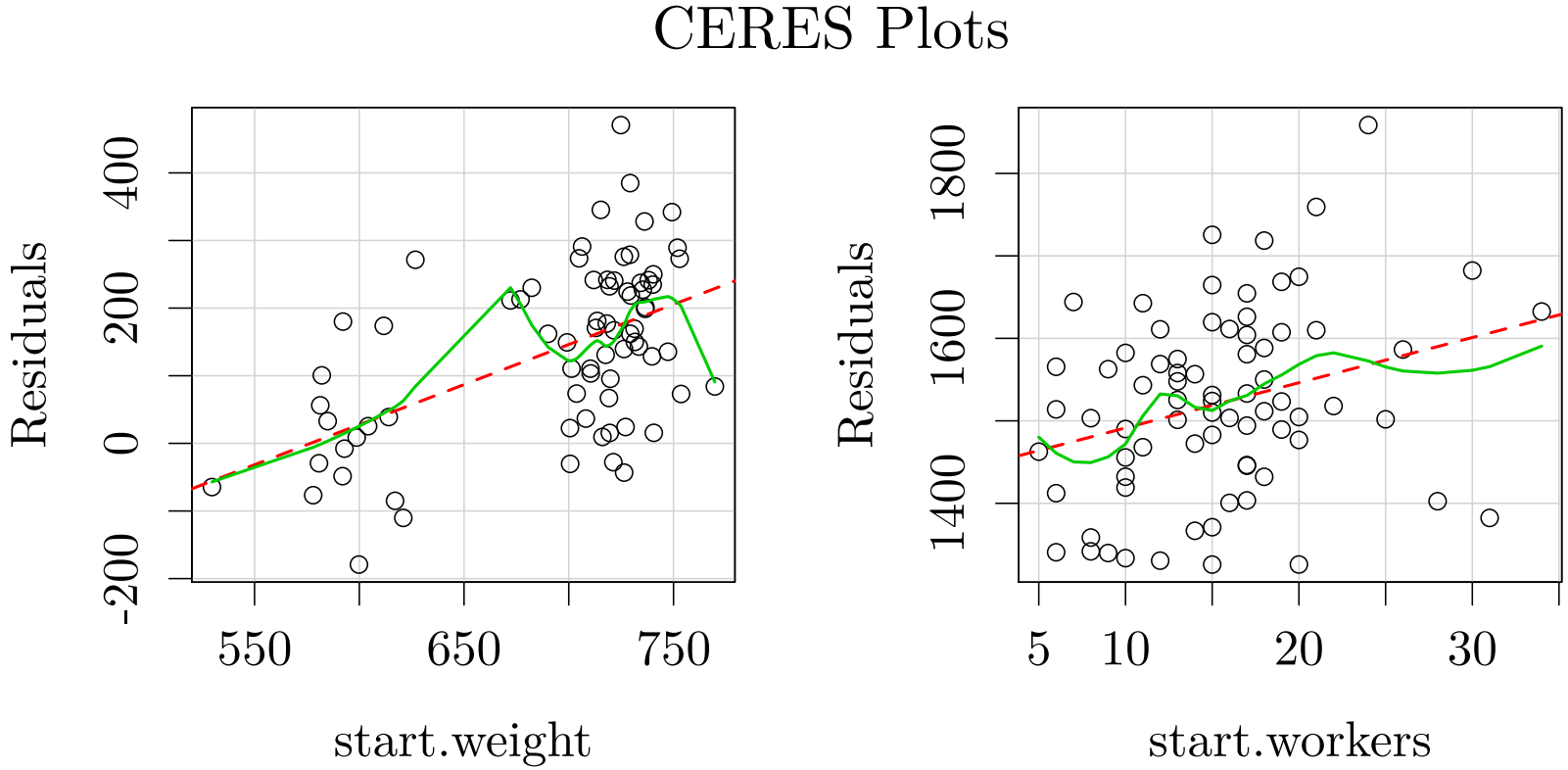
$$PR[x_{i3}] = y_i - (\alpha + \beta_{i1}x_{i1} + \beta_{i2}x_{i2})$$

Relationship should be approximately linear (conform to fitted line).

# Diagnostics for linear models

CERES plots:

```
car::ceresPlots(weight_lm, ylab = "Residuals")
```




Red line is fitted line, green line is smoothed relationship

# Diagnostics for linear models

Assumption 4: **Independence**

If residuals are non-independent (eg. correlated), they are not accurately modelled by a normal linear model.

**Best defense**: a well thought-out sampling scheme.

**To detect**: plot residuals vs. spatial, temporal covariates.

**If present** (intentionally or not): hierarchical or mixed models can model complex dependence structures.

# Collinearity

If two covariates are **highly correlated** aka **collinear** $\Rightarrow$ cannot statistically distinguish which one influences the response.

Covariate 1 could have large coefficient & covariate 2 could have small coefficient, or vis versa.

$$y = 2x_1 + 4x_2 = 4x_1 + 2x_2 \text{ if } x_1 = x_2$$

Many values are equally likely for the regression coefficients.

Estimated standard errors for coefficients **will be inflated**, compared to model where only one is present.

# Collinearity

The **variance inflation factor** (VIF) is a measure of how much the estimated standard error of a regression coefficient is increased by collinearity.

Use **vif(<lm object>)** in **car** package:

```
car::vif(weight_lm)

##                GVIF Df GVIF^(1/(2*Df))
## start.weight  1.008  1           1.004
## start.workers 1.098  1           1.048
## treatment     1.095  2           1.023
```

Example of a collinear model:

```
collinear_model <- lm(reproductives ~ max.weight + range.weight,
    data = Bombus)
car::vif(collinear_model)

##   max.weight range.weight
##        5.791        5.791
```

# Diagnostics for linear models

**Rule of thumb**: if square root of VIF is greater than 2, be concerned.

```
sqrt(car::vif(collinear_model)) > 2
```

```
##   max.weight range.weight
##         TRUE         TRUE
```

Can also use **cov2cor(vcov(<lm object>))** to view correlation matrix among regression terms.

**If collinearity is a problem**:

- Use biological insight to remove one of the offending covariates
- Use dimension reduction (i.e. PCA) to collapse into orthogonal variables

# Diagnostics for linear models

**Outlier detection**: outliers are observations far from fitted value.

Use studentized residuals vs. fitted values to detect:

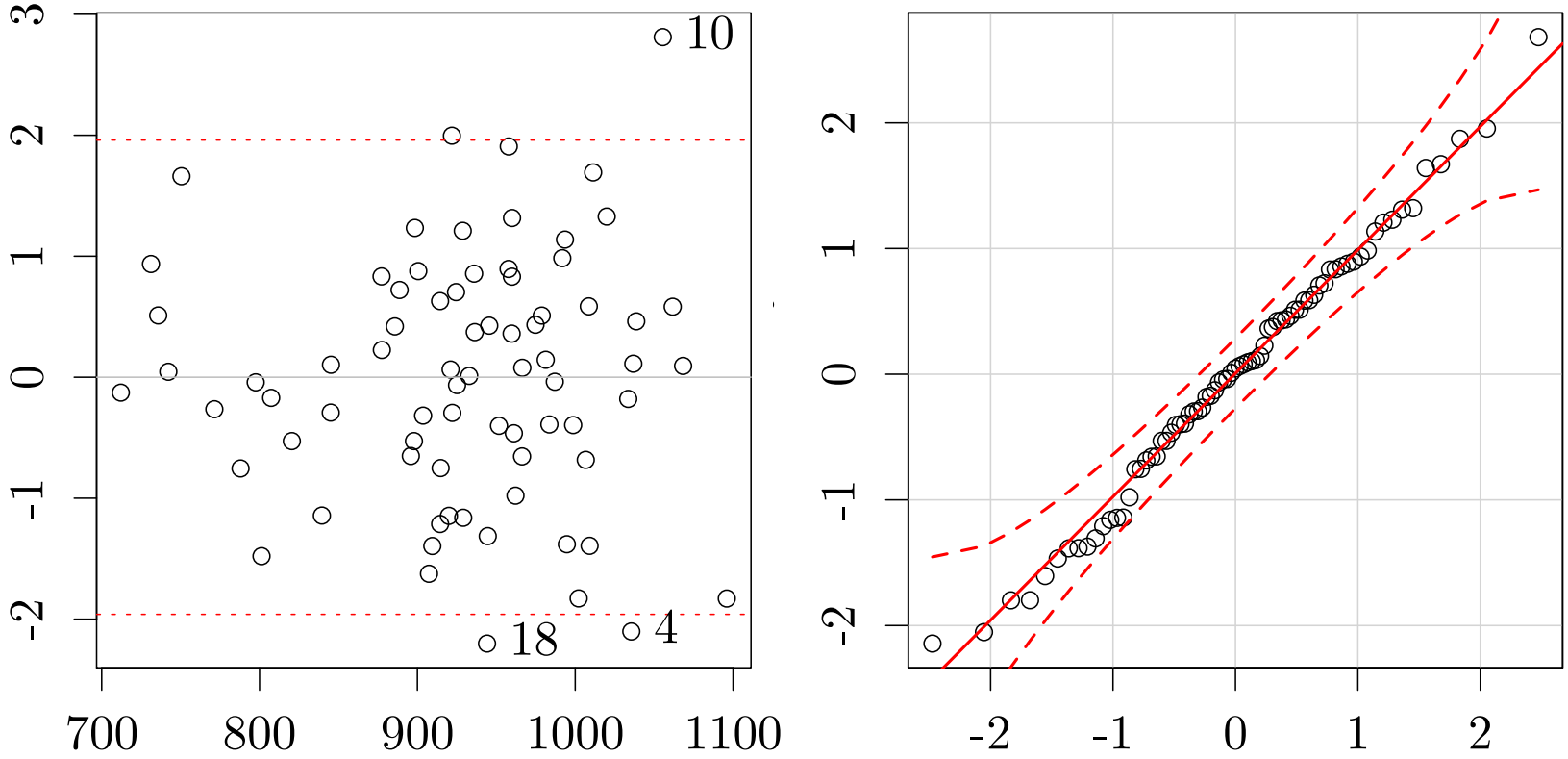
```
# plot studentized residuals
plot(fitted(weight_lm), rstudent(weight_lm),
    xlab = "Fitted Values", ylab = "Studentized Residuals")
abline(h = c(0, -1.96, 1.96), col = c("gray",
    "red", "red"), lty = c(1, 3, 3))
# detect and label outliers
is_outlier <- which(abs(rstudent(weight_lm)) >
    2)
text(fitted(weight_lm)[is_outlier], rstudent(weight_lm)[is_outlier],
    labels = is_outlier, pos = 4)
```

Also look for outliers in Q-Q plot:

```
car::qqPlot(rstandard(weight_lm), ylab = "Observed Quantiles",
    xlab = "Theoretical Quantiles")
```

# Diagnostics for linear models

Labelled points (beyond 2 standard deviations) are past 0.95th quantile.




Two questions to ask once outliers are detected:

- Do they influence our results (fitting, hypothesis testing)?
- Why are they not well fit (is there some process we are not modelling)?

# Diagnostics for linear models

**Influence measures**: **influential** points are observations that highly influence the estimates of the regression coefficients.

These are **not necessarily** outliers!

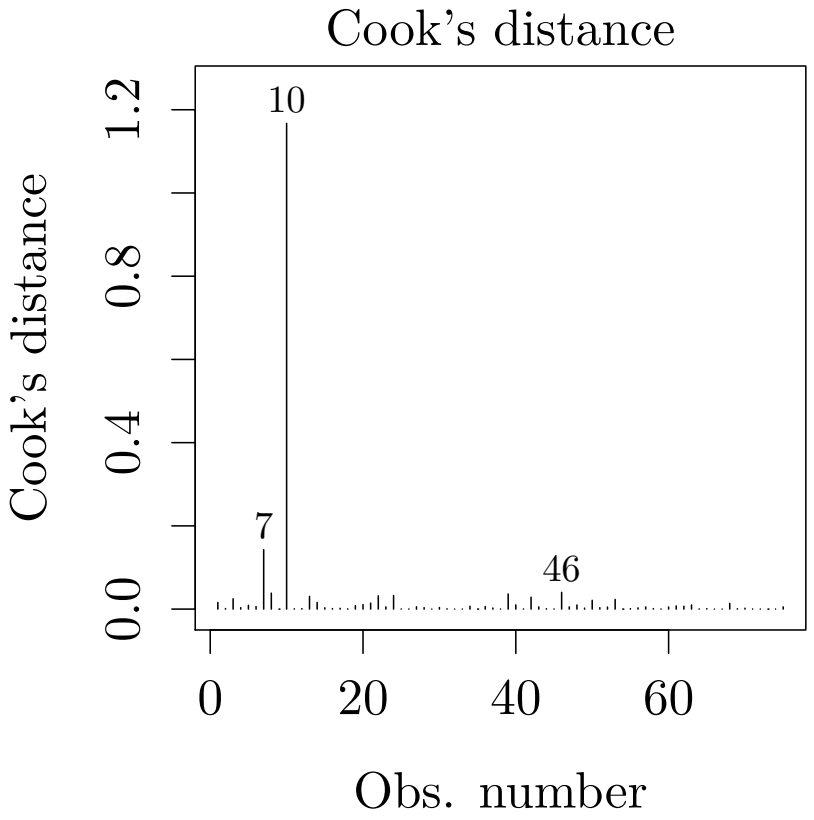
Use **influence.measures(<lm object>)** to calculate influence measures and flag influential observations.

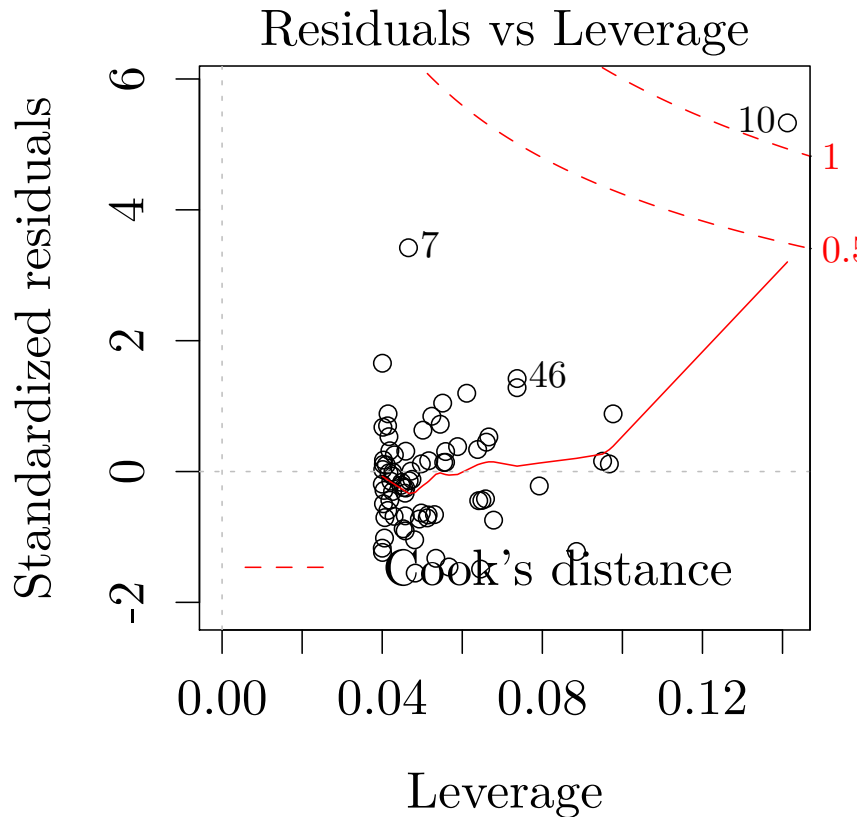
```
influence.measures(weight_lm)$infmat[1:3, 1:5]  # influence measures

##      dfb.1_ dfb.strt.wg dfb.strt.wr dfb.trtH dfb.trtL
## 1  0.003642    0.003821    -0.01374  -0.0238 -0.02287
## 2 -0.007485   -0.057954     0.10134   0.2188  0.21307
## 3  0.016959    0.018572    -0.01461  -0.1608 -0.16340

influence.measures(weight_lm)$is.inf[1:3, 1:5]  # flagged 'TRUE' if influential

##   dfb.1_ dfb.strt.wg dfb.strt.wr dfb.trtH dfb.trtL
## 1  FALSE       FALSE       FALSE    FALSE    FALSE
## 2  FALSE       FALSE       FALSE    FALSE    FALSE
## 3  FALSE       FALSE       FALSE    FALSE    FALSE
```

# Diagnostics for linear models

**Cook's Distance**: combines outlying-ness with influence.

```
plot(reprod_lm, which = 4)
plot(reprod_lm, which = 5)
```




Points beyond red dashed boundary have large Cook's distance

# Prediction

**Prediction**: calculate expected value of response for arbitrary values of covariates. Basic formula: generate data frame with covariate values to predict for,

```
to_predict = expand.grid(start.weight = c(200,
    400, 600), treatment = c("Low", "High",
    "Control"), start.workers = 20)
```

Then pass to **predict(<lm object>)**:

```
predicted_response <- predict(weight_lm,
    newdata = to_predict, se.fit = T)  # the predicted response
head(predicted_response$fit)
```

```
##     1     2     3     4     5     6
## 352.6 591.6 830.6 329.0 568.0 807.1
```

Argument "**se.fit = T**" for standard errors of predictions.

# Hypothesis testing

Hypothesis testing discussed here is **null hypothesis testing**.

1. Determine probability distribution of null hypothesis for some test statistic.
2. Calculate probability of a test statistic *larger than the observed test statistic*, under the null hypothesis.
3. If this probability is lower than arbitrary value (*significance level*), have evidence to reject the null hypothesis.

**Null hypotheses can be pretty trivial** (just because you reject a null hypothesis don't think that you have found something important).

# Hypothesis testing

Tests of single coefficients in **summary(<lm object>)**

```
summary(weight_lm)$coef[1, ]

##   Estimate Std. Error    t value   Pr(>|t|)
##    58.4888   166.7101     0.3508     0.7268
```

**Null hypothesis**: coefficient = 0

**Test statistic**: T-value = ratio of the coefficient and its standard error. For example, for (Intercept):

$$T_{\text{intercept}} = \frac{58.49}{166.71} = 0.351$$

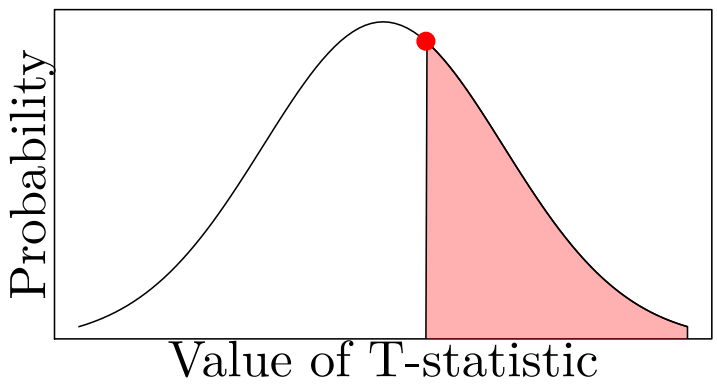
MLE and standard error of coefficient are random variables, so ratio is also random. Null T distribution is centered at 0, with (degrees freedom) = (data points) - (number of parameters).

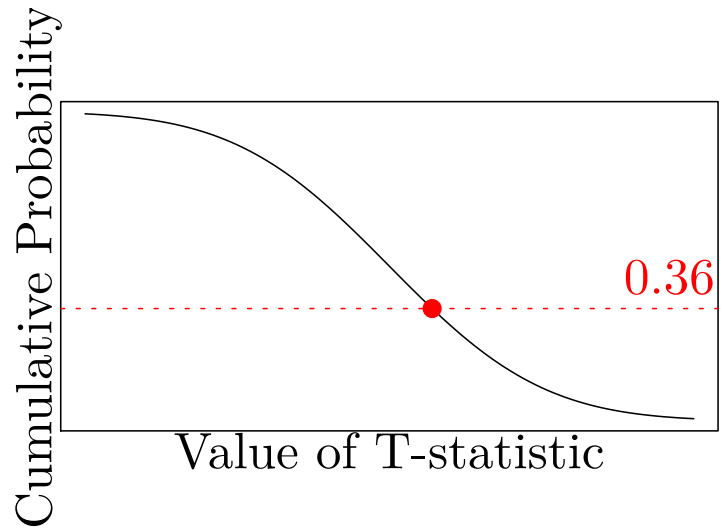
# Hypothesis testing

Given out observed T statistic, we ask:

- What is probability that a random T-statistic from a central T distribution with $75 - 5 = 70$ degrees freedom ...
- Is higher than our observed T statistic?

We use the **cumulative distribution function** (CDF) of T distribution to find this:




Two-tailed probability of $T_{\text{intercept}} = 0$: $0.363 \cdot 2 = 0.726$

# Hypothesis testing

**Confidence interval**: if an experiment is repeated 100 times, and a $P\%$ confidence interval is constructed each time for a parameter of interest, $P\%$ of the confidence intervals will include the true value of the parameter. Formula:

$$0.95\text{CI} = \text{Mean} \pm (0.95 \text{ Quantile} \cdot \text{Standard Error})$$

In R, use **confint(<lm object>)** which employs 0.95 quantile of standard T distribution.

```
confint(weight_lm)
```

```
##                    2.5 %  97.5 %
## (Intercept)   -274.0040 390.982
## start.weight     0.7451   1.645
## start.workers    0.9121   9.977
## treatmentHigh -142.8780 -11.894
## treatmentLow  -117.6039   9.941
```

# Hypothesis testing

**Tests of main effects**: Is the increase in explanatory power associated adding an **'effect'** (such as **Treatment**) substantial, given the increase in model complexity. $\Rightarrow$ Think of this as model selection.

Imagine a standard measure, such as: $\frac{\text{Improvement in fit}}{\text{Increase in complexity}}$

If we define a null model and a full model–*the full model has all covariates*–could calculate this for each model in between.

# Hypothesis testing

If 'Improvement in fit' depends on the scale of the data ... then we need a reference measure (defined against full model) to compare with, to get scale-less standard measure:

$$\frac{\text{Room for improvement in fit}}{\text{Room to add complexity}}$$

We need two metrics: one for **improvement in fit**, another for **increase in complexity**.

# Hypothesis testing

Let $\bar{y}$ be the mean of the response.

Let $\mathbb{E}[y_i]_{model}$ be the fitted value for the $i$th response for a given model.

Define **total sum of squares** as :

$$SS_{tot} = \sum_{i=0}^{N} (y_i - \bar{y})^2$$

Define **regression sum of squares** for a given regression model as :

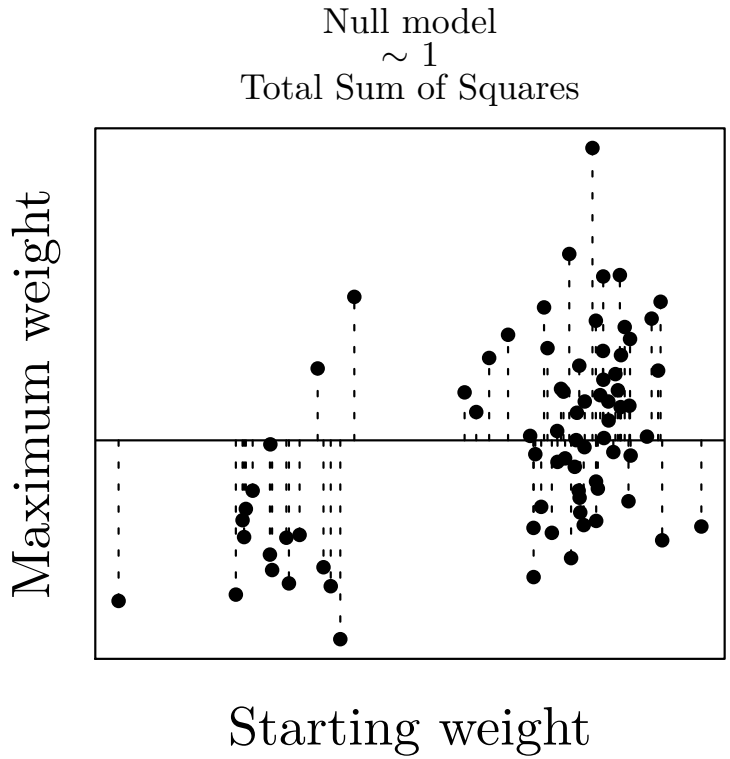
$$SS_{reg} = \sum_{i=0}^{N} (\mathbb{E}[y_i]_{model} - \bar{y})^2$$

Define **residual sum of squares** as :

$$SS_{res} = \sum_{i=0}^{N} (y_i - \mathbb{E}[y_i]_{model})^2$$

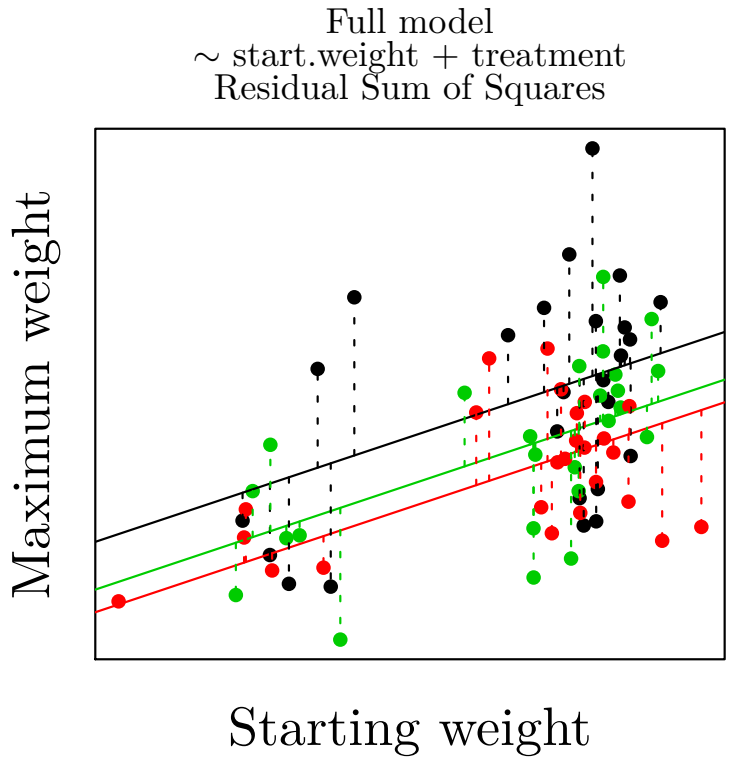
# Hypothesis testing

The **total sum of squares** is the sum of squares of a 'null' model – a model with just an intercept (aka mean parameter).



# Hypothesis testing

The **residual sum of squares** (RSS) is the sum of squares of the 'full' model.




and has **residual degrees freedom** $N - p_{max}$.

# Hypothesis testing

The **residual sum of squares** and **residual degrees freedom** combine to form our "reference measure" aka **residual mean square**.

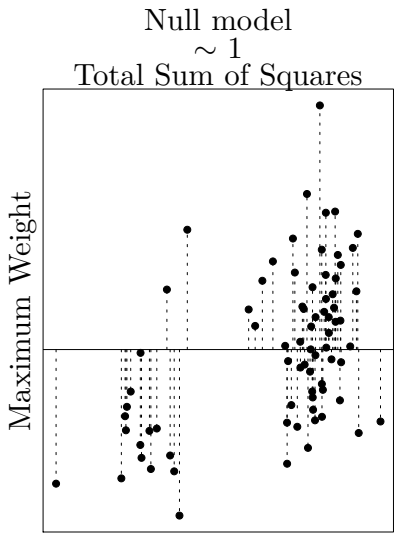
$$\frac{\text{Room for improvement in fit}}{\text{Room to add complexity}} = \frac{\sum_{i=0}^{N}(y_i - \mathbb{E}[y_i]_{model})^2}{N - p_{max}}$$

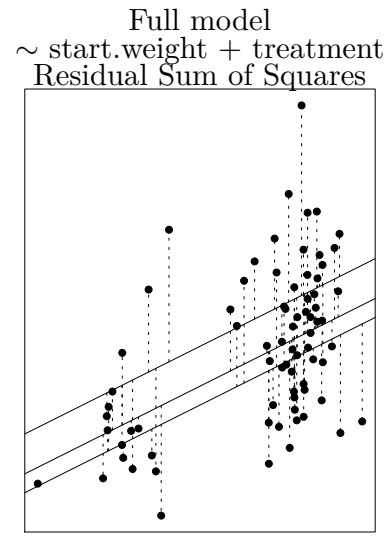
We add terms to model, calculating increase in the regression sums of squares $\Delta SS$, and increase in parameter number $\Delta p$.

With each additional term, calculate our "standard measure" aka **mean square** $= \frac{\Delta SS_{reg}}{\Delta p}$

# Hypothesis testing

Begin with **null** model and **full** model, calculate **total** and **residual** sum of squares and associate degrees freedom

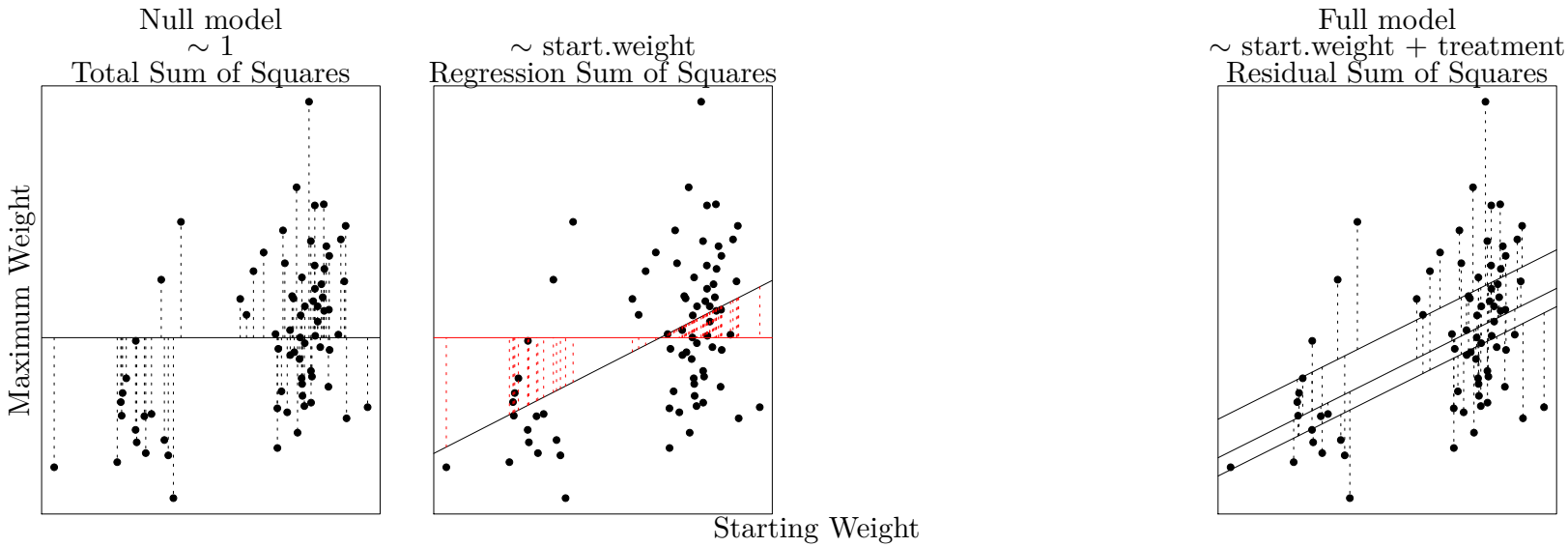



Starting Weight

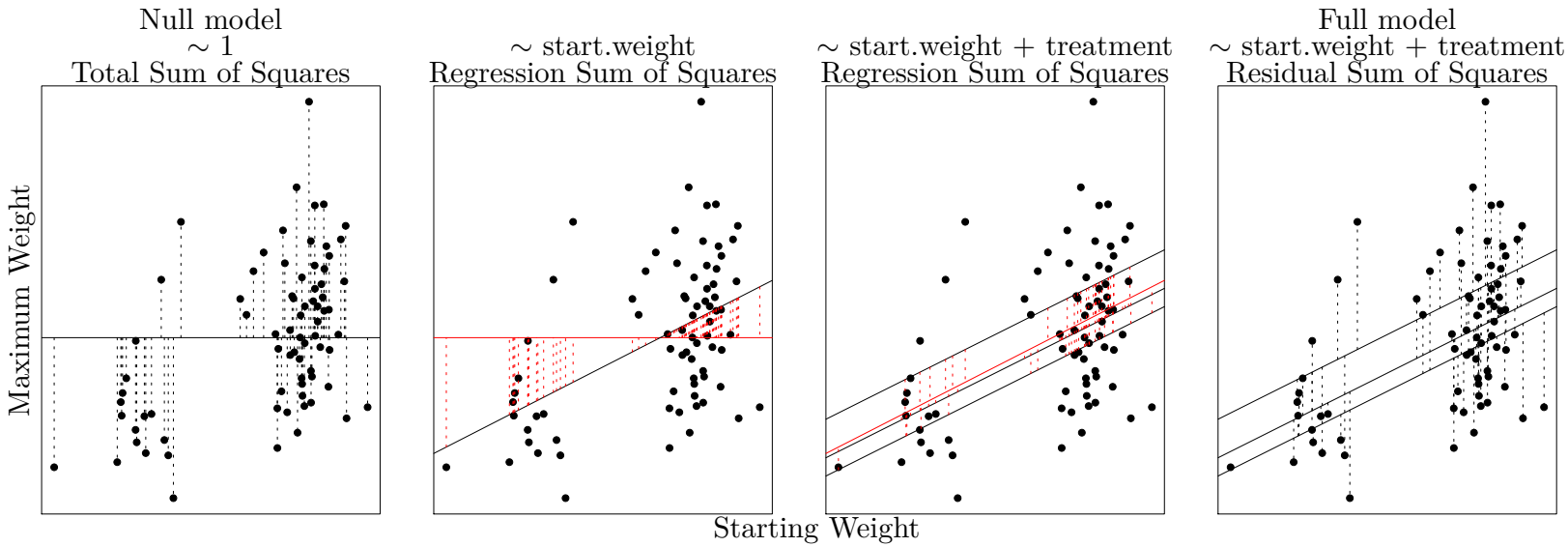
# Hypothesis testing

Add covariate to null model. Calculate **regression sums of squares** (improvement in fit) and the increase in parameter number (increase in complexity).



# Hypothesis testing

Add LAST covariate to previous model. Calculate NEW **regression sums of squares** (improvement in fit) and the increase in parameter number (increase in complexity).



# Hypothesis testing

The **F-statistic** is ratio of regression mean square to residual mean square (at each step) – this is our scale-less standard measure.

$$F_{SS} = \frac{\left(\frac{SS_{reg}}{\Delta p}\right)}{\left(\frac{SS_{res}}{n - p_{max}}\right)}$$

Null distribution is **F-distribution** with $\Delta p$ and $N - p_{max}$ degrees freedom.

Calculate probability that a random F-statistic from null distribution, is greater than observed F-statistic.

# Hypothesis testing

This is **sequential** aka **"Type I"** sums of squares. In R, use **anova(<lm object>)**.

```
anova(weight_lm)
```

```
## Analysis of Variance Table
##
## Response: max.weight
##                Df Sum Sq Mean Sq F value  Pr(>F)
## start.weight    1 338535  338535   27.45 1.6e-06 ***
## start.workers   1 129584  129584   10.51  0.0018 **
## treatment       2  71902   35951    2.92  0.0608 .
## Residuals      70 863225   12332
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Sensitive to ordering of effects.

# Hypothesis testing

For **marginal** aka **"Type III"** sums of squares use **car::Anova(<lm object>)**.

```
car::Anova(weight_lm, type = "3")

## Anova Table (Type III tests)
##
## Response: max.weight
##                Sum Sq Df F value  Pr(>F)
## (Intercept)      1518  1    0.12   0.727
## start.weight   345998  1   28.06 1.3e-06 ***
## start.workers   70781  1    5.74   0.019 *
## treatment       71902  2    2.92   0.061 .
## Residuals      863225 70
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Not sensitive to ordering of effects (but many statisticians don't like it)

# Hypothesis testing

The **coefficient of variation** aka $R^2$ is the proportion of variance explained by the model.

$$R^2 = 1 - \frac{SS_{residual}}{SS_{total}}$$

An **omnibus test** asks how well the model as whole explains the data relative to a null model (ie. intercept only).

We can perform an omnibus F test on $R^2$ by defining the F statistic:

$$F_{R^2} = \frac{\left(\frac{R^2}{p_{max}-1}\right)}{\left(\frac{1-R^2}{n-p_{max}}\right)}$$

With null distribution as F-distribution with $p_{max} - 1$ and $n - p_{max}$ degrees freedom.

# Hypothesis testing

This is given at the end of **summary(<lm object>)**:

```
summary(weight_lm)


##
## Call:
## lm(formula = max.weight ~ start.weight + start.workers + treatment,
##     data = Bombus)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -222.53  -70.08    4.79   72.56  288.54
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)     58.489    166.710    0.35    0.727
## start.weight     1.195      0.226    5.30  1.3e-06 ***
## start.workers    5.445      2.273    2.40    0.019 *
## treatmentHigh  -77.386     32.837   -2.36    0.021 *
## treatmentLow   -53.831     31.975   -1.68    0.097 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 111 on 70 degrees of freedom
## Multiple R-squared:  0.385, Adjusted R-squared:  0.35
## F-statistic: 10.9 on 4 and 70 DF,  p-value: 5.96e-07
```

# Multiple comparisons

When we do a set of all pairwise comparisons (ie. among treatment groups), we run into the multiple comparisons problem.

Essentially, our comparisons are not independent, and we inflate our Type-I error (probability to detect effect when there is none).

Many solutions to control for inflation, most common is probably Tukey's Honest Significant Difference (HSD).

# Multiple comparisons

In R:

```
tukey_weight_lm <- multcomp::glht(weight_lm, linfct = multcomp::mcp(treatment = "Tukey"))
summary(tukey_weight_lm)


##
##   Simultaneous Tests for General Linear Hypotheses
##
## Multiple Comparisons of Means: Tukey Contrasts
##
##
## Fit: lm(formula = max.weight ~ start.weight + start.workers + treatment,
##     data = Bombus)
##
## Linear Hypotheses:
##                       Estimate Std. Error t value Pr(>|t|)
## High - Control == 0    -77.4       32.8   -2.36    0.055 .
## Low - Control == 0     -53.8       32.0   -1.68    0.219
## Low - High == 0         23.6       31.6    0.74    0.738
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
## (Adjusted p values reported -- single-step method)
```

There is also **TukeyHSD(anova(<model name>))** in base R, but function **glht()** extends to complex comparisons.

# Log-Transformation

**Transformations** (often log) used to "normalize" response.

This is not a good move for count or proportion data – use a GLM.

However, for continuous all-positive data, this is OK – and is equivalent to fitting **log-normal** model.

Sometimes we might want to compare log-normal model to equivalent normal model. But in R, the log-likelihoods are on different scales:

```
norm_lm <- lm(max.weight ~ start.weight, data = Bombus)  # normal model
lognorm_lm <- lm(log(max.weight) ~ start.weight, data = Bombus)  # log-normal model
c(logLik(norm_lm), logLik(lognorm_lm), AIC(norm_lm), AIC(lognorm_lm))
```

```
## [1] -464.95   48.77  935.90  -91.55
```

# Log-Transformation

**Solution**: calculate likelihood using **log-normal pdf** (**dlnorm()** in R).

```
lognorm_sigma <- summary(lognorm_lm)$sigma
lognorm_mu <- fitted(lognorm_lm)
lognorm_np <- lognorm_lm$rank
lognorm_logLik <- sum(dlnorm(Bombus$max.weight, meanlog = lognorm_mu,
    sdlog = lognorm_sigma, log = T))
lognorm_AIC <- -2 * lognorm_logLik + 2 * lognorm_np  # AIC
c(AIC(norm_lm), lognorm_AIC)

## [1] 935.9 929.8
```

# Parametric bootstrapping / Simulation

General idea is to see if observed data are consistant with model, when analytical methods are not very clear:

1. Have model, have observed data
2. Simulate data from model
3. Calculate some statistic from simulated data (possibly refiting model to simulated data)
4. Simulations of statistic give estimated distribution of that statistic
5. Check consistency of observed statistic, against simulated distribution of statistic

Really very useful for complex linear models (not the ones here). Some Mickey-Mouse examples in the script.