

Программирование и статистический анализ данных на языке R

Лекция 5 (Основы статистического
анализа на языке R)

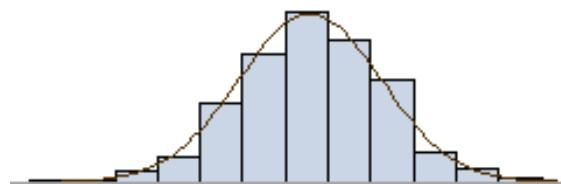


Петровский Михаил (ВМК МГУ), michael@cs.msu.su

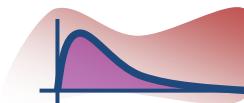
Основные предположения линейной регрессии

- Независимость наблюдений (и ошибок)
- Нормальное распределение ошибки с константной дисперсией

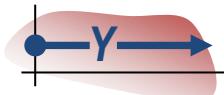
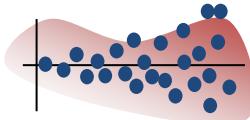
$$\varepsilon \sim iid N(0, \sigma^2)$$



- Часто возникающие «особенности»:
 - Несимметричные распределения отклика



- Гетероскедастичность
- Ограниченнная область определения отклика



- Что делать?
 - Явно преобразовывать отклик:
 - Использовать функцию связи:

$E(g(Y) | X)$

$g(E(Y | X))$

Преобразование отклика и логнормальная регрессия

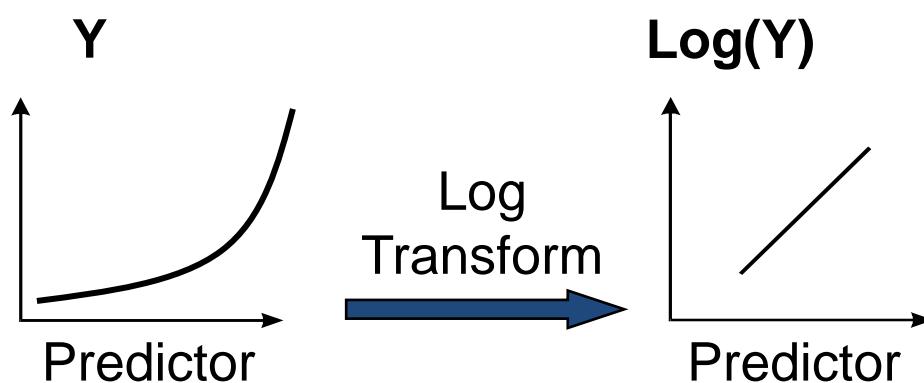
- Если логнормальное распределение отклика Y , тогда $\log(Y)$ – нормальное

$$\log(Y) \sim N(\mu, \sigma^2)$$

$$E[Y] = \exp\left(\mu + \frac{\sigma^2}{2}\right)$$

$$Var[Y] = (e^{\sigma^2} - 1)(E[Y])^2$$

- Строим модель для преобразованного отклика:



$$E[\log(Y)] = X\hat{\beta}$$

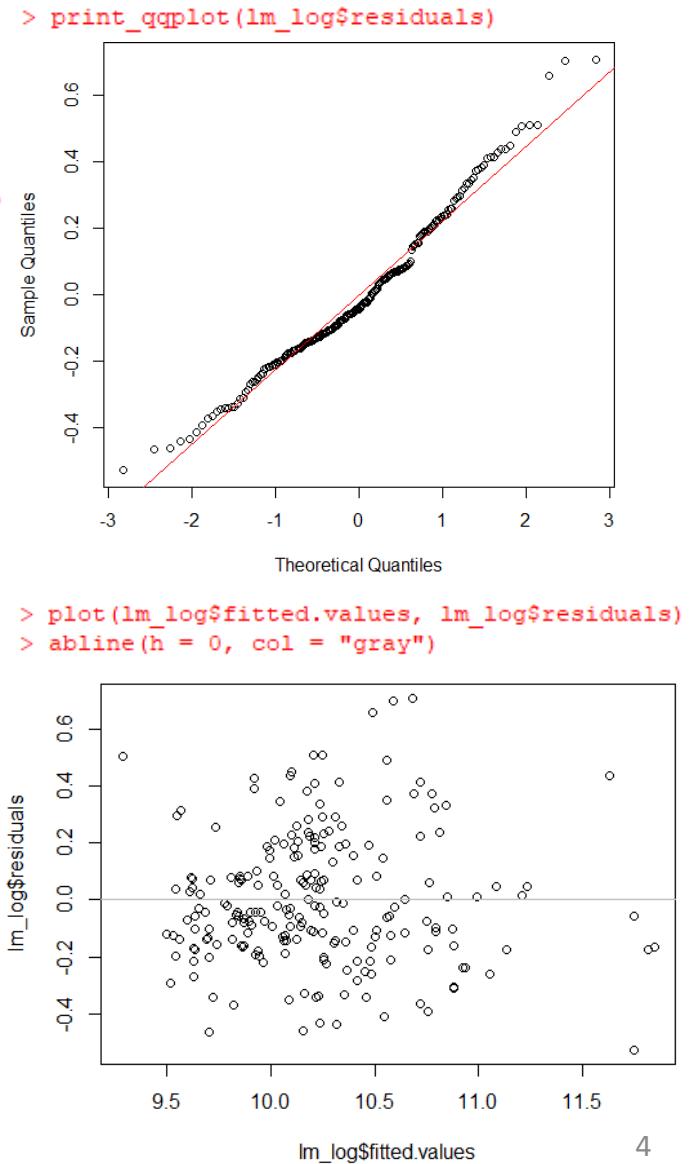
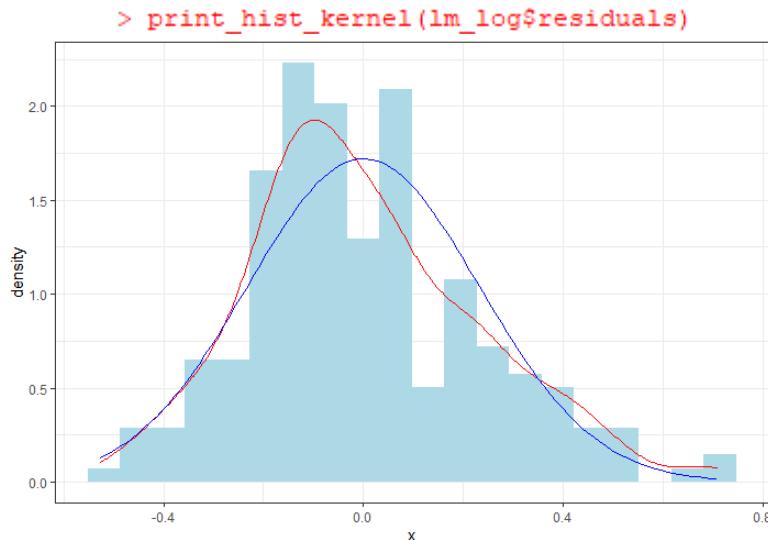
Но чему равно $E[Y]$?

$$E[Y] \approx \exp\left(X\hat{\beta} + \frac{\sigma^2}{2}\right)$$

MSE, но
желательно на
валидационном
наборе

Пример логнормальной регрессии

```
> smp_size <- nrow(cars) / 2
> train_ind <- sample(seq_len(nrow(cars)), size = smp_size)
>
> train <- cars[train_ind, ]
> test <- cars[-train_ind, ]
>
> lm_log <- lm(log(Invoice + 1) ~ Wheelbase + Weight + Horsepower, train)
> p_log_Invoice <- predict(lm_log, test)
> mse <- sum((p_log_Invoice - log(test$Invoice + 1))^2) / nrow(cars)
>
> cars$p_Invoice <- exp(predict(lm_log, cars) + mse/2) - 1
> cars$Car <- paste(cars$Make, cars$Model)
> head(cars[,c("Car", "Invoice", "p_Invoice")])
# A tibble: 6 × 3
  Car           Invoice p_Invoice
  <chr>        <dbl>     <dbl>
1 Acura MDX      33337    40808.
2 Acura RSX Type S 2dr  21761    23455.
3 Acura TSX 4dr    24647    24316.
4 Acura TL 4dr     30299    36193.
5 Acura 3.5 RL 4dr  39014    28269.
6 Acura 3.5 RL w/Navigation 4dr 41100    28324.
```

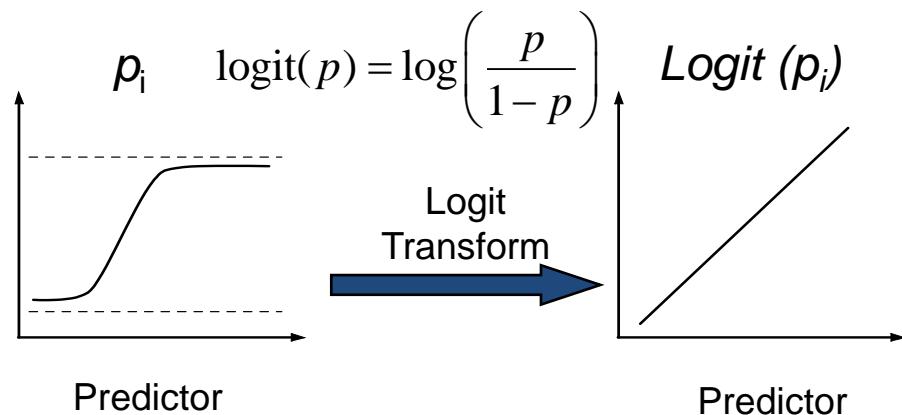


Если не нормальное распределение ошибки?

Функция связи

$$g(E(y_i)) = \beta_0 + \beta_1 x_{1i} + \dots + \beta_k x_{ki} = X\beta$$

- Распределение отклика наблюдений принадлежит экспоненциальному семейству. $f(y | \theta) = h(y)c(\theta)\exp(t(y) \cdot W(\theta))$
- Дисперсия зависимой переменной Y – функция от среднего.
- $X\beta$ моделирует функцию от $E(y)$ (link function – функция связи)
- Распределение отклика наблюдений может подсказать какую функцию связи выбрать (далее)
- Пример (лоистическая регрессия):



$$f(y | p) = p^y (1-p)^{1-y} = \\ = (1-p)I_y \exp(y \log(p/(1-p)))$$

$$I_y = \begin{cases} 1, & y \in \{0,1\} \\ 0, & \text{иначе} \end{cases}$$

Типовые функции связи для обобщенных линейных моделей

Model	Response	Distribution	Mean	Variance	Canonical Link
Linear Regression	Continuous	Normal	μ	σ^2	identity μ
Logistic regression	Dichotomous	Binomial	π	$\pi(1 - \pi)/n$	logit $\log[\pi/(1-\pi)]$
Poisson Regression	Count	Poisson	λ	λ	log $\log(\lambda)$
Gamma Regression	Continuous	Gamma	μ	μ^2/ν	*inverse $1/\mu$

*часто используется функция связи LOG

Параметры положения и разброса

Экспоненциальное семейство распределений: $f(y | \theta) = \exp\left\{\frac{y\theta - b(\theta)}{a(\phi)} + c(y, \phi)\right\}$

- Линейная регрессия

$$f(y | \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-\frac{(y-\mu)^2}{2\sigma^2}},$$

- Логистическая регрессия

$$f(y | p) = p^y \cdot (1-p)^{(1-y)}$$

- Пуассоновская регрессия

$$f(y | \lambda) = \frac{e^{-\lambda} \lambda^y}{y!}$$

- Гамма регрессия

$$f(y | \nu, \mu) = \frac{1}{\Gamma(\nu) \cdot y} \cdot \left(\frac{y\nu}{\mu}\right)^\nu \cdot e^{-\left(\frac{y}{\mu}\right)},$$

Регрессия	Параметр положения	Параметр разброса
Линейная	μ	σ
Логистическая	p	1
Пуассоновская	λ	1
Гамма	μ	ν

Процедура GLM

```
glm(formula, family = gaussian, data, weights, subset,  
na.action, start = NULL, etastart, mustart, offset,  
control = list(...), model = TRUE, method = "glm.fit",  
x = FALSE, y = TRUE, singular.ok = TRUE, contrasts = NULL, ...)  
glm.fit(x, y, weights = rep(1, nobs),  
start = NULL, etastart = NULL, mustart = NULL,  
offset = rep(0, nobs), family = gaussian(),  
control = list(), intercept = TRUE, singular.ok = TRUE)
```

- Основные настройки через параметры модели:
 - family= распределение ошибки (gaussian, binomial, Gamma, inverse.gaussian, quasi ...)
 - функция связи (в том числе кастомизированная) может задаваться в параметре link в family (log, logit, probit, identity, ...)
 - много дополнительных параметров и настроек (читайте манул), например, можно задавать параметры разброса

Оценка отклонения

Поиск параметров модели

- решается задача оптимизации
- max loglik с заданным распределением и функцией связи

Распределение Отклонение

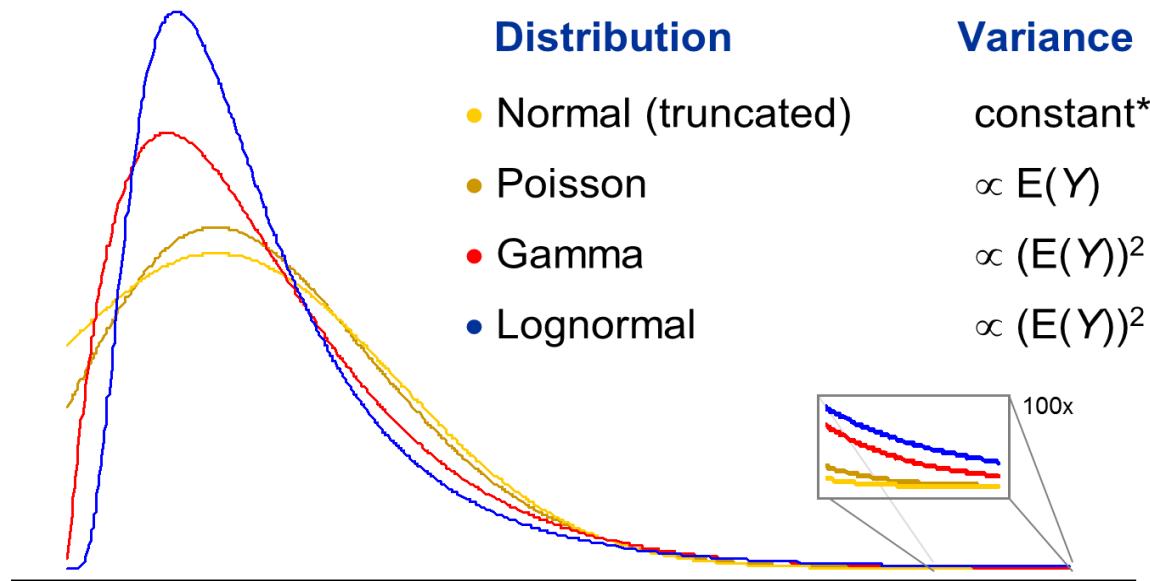
Normal $Q(\mathbf{w}) = \sum (y - \mu(\mathbf{w}))^2$

Poisson $Q(\mathbf{w}) = 2 \sum [y \ln(y / \mu(\mathbf{w})) - (y - \mu(\mathbf{w}))]$

Gamma $Q(\mathbf{w}) = 2 \sum [-\ln(y / \mu(\mathbf{w})) + (y - \mu(\mathbf{w})) / \mu(\mathbf{w})]$

Bernoulli $Q(\mathbf{w}) = -2 \sum [y \ln(\mu(\mathbf{w})) + (1 - y) \ln(1 - \mu(\mathbf{w}))]$

Выбор распределения для обобщенной линейной модели



- В случае гетероскедастичности вместо линейной часто применяется гамма регрессия (с различными функциями связи)
- Гамма распределение:
 - Асимметричное распределение для **положительных** значений
 - Дисперсия пропорциональна квадрату среднего
 - Хвост «легче» чем у логнормального

Пример оценки распределения отклика

```
> feature <- cars$Invoice  
>  
> scale_gamma <- var(feature)/mean(feature)  
> shape_gamma <- mean(feature)^2/var(feature)  
>  
> x_r <- rgamma(length(feature), scale = scale_gamma, shape = shape_gamma)
```

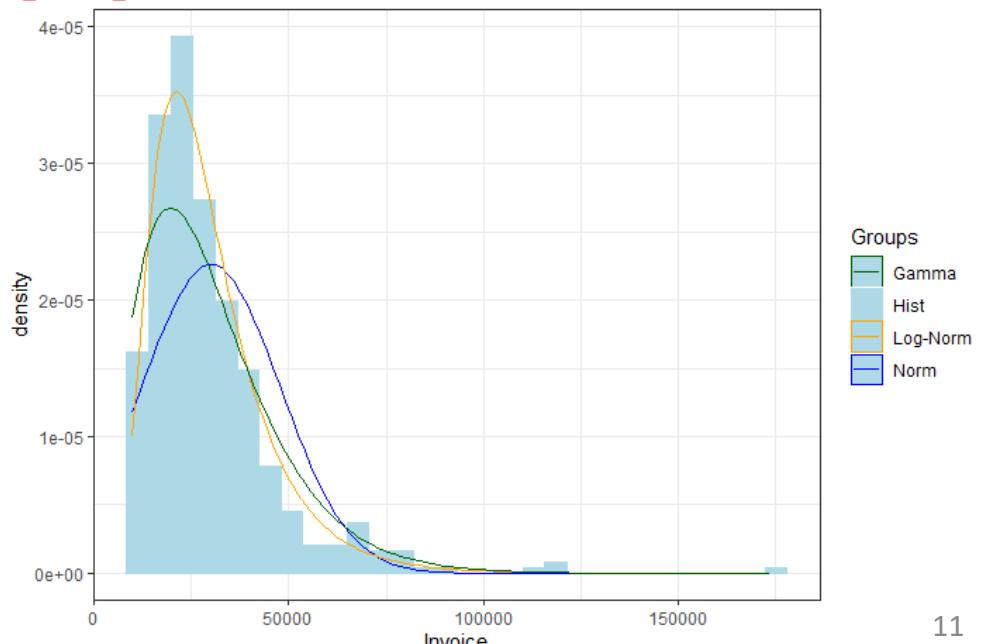
Normal

```
> ks.test(feature, "pnorm", mean(feature),  
          sd(feature), alternative = "gr")  
data: feature  
D^+ = 0.14093, p-value = 4.136e-08  
> cvm.test(feature)  
data: feature  
W = 3.4519, p-value = 7.37e-10  
> ad.test(feature)  
data: feature  
A = 20.395, p-value < 2.2e-16
```

```
> x <- cars$Invoice  
> dnorm_pars <- c(mean = mean(x), sd = sd(x))  
> dlnorm_pars <- c(mean = mean(log(x)), sd = sd(log(x)))  
> dgamma_pars <- c(scale = var(x)/mean(x), shape = mean(x)^2/var(x))  
>  
> ggplot(cars, aes(x = Invoice)) +  
+ geom_histogram(aes(y = ..density.., colour = "Hist"), fill="light blue", bins=30) +  
+ stat_function(fun = dnorm, aes(colour = "Norm"), args = dnorm_pars) +  
+ stat_function(fun = dlnorm, aes(colour = "Log-Norm"), args = dlnorm_pars) +  
+ stat_function(fun = dgamma, aes(colour = "Gamma"), args = dgamma_pars) +  
+ theme_bw() +  
+ scale_colour_manual("Groups", values = c("dark green", "light blue", "orange", "blue"))
```

Log-Normal

```
> ks.test(feature, "plnorm", mean(log(feature)),  
          sd(log(feature)), alternative = "gr")  
data: feature  
D^+ = 0.041615, p-value = 0.2271  
> cvm.test(log(feature))  
data: log(feature)  
W = 0.15855, p-value = 0.0183  
> ad.test(log(feature))  
data: log(feature)  
A = 1.1894, p-value = 0.004161
```



Gamma

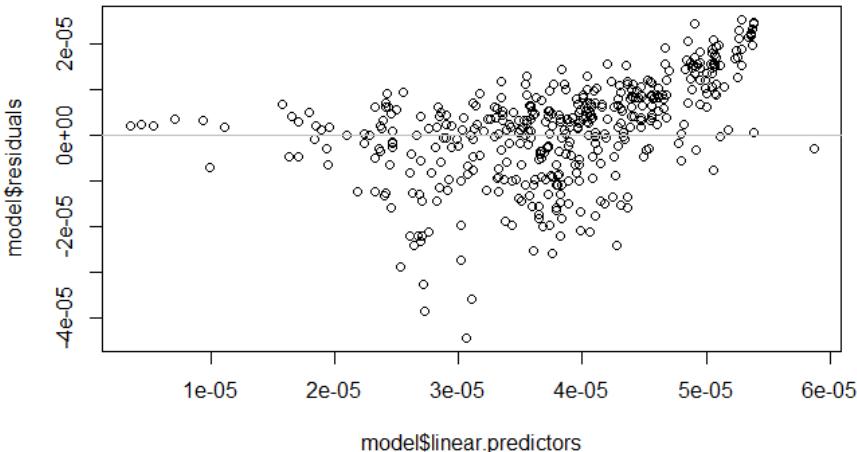
```
> ks.test(feature, "pgamma", scale = scale_gamma,  
          shape = shape_gamma, alternative = "gr")  
data: feature  
D^+ = 0.070772, p-value = 0.01374  
> cvm.test(x_r)  
data: x_r  
W = 0.81011, p-value = 1.222e-08  
> ad.test(x_r)  
data: x_r  
A = 5.1923, p-value = 7.654e-13
```

Пример гамма регрессии

```

> full_summary <- function(model, df){
+   print(summary(model))
+
+   crit_gof <- glance(model) [, -c(1, 2, 8)]
+   crit_gof$R2 <- R2(model$fitted, model$y, "traditional")
+   crit_gof$RMSE <- RMSE(model$fitted, model$y)
+   crit_gof$Value_DF <- deviance / model$df.residual
+   crit_gof$Chi_DF <- sum(residuals(model, type = "pearson",
+                                     weights=cars_counts$Models)^2) / model$df.residual
+   print(crit_gof)
+
+   plot(model$linear.predictors,
+        model$residuals)
+   abline(h = 0, col = "gray")
+
+   print(cbind(tidy(model) [, -1], ci = confint(model)))
+ }
>
> plot_predict <- function(model, df){
+   res <- data.frame(
+     Car = paste(df$Make, df$Model),
+     Invoice = df$Invoice,
+     p_Invoice = 1/predict(model, df)
+   )
+   head(res)
+ }

```



```

> model_gamma <- glm(Invoice ~ Weight + Wheelbase + Horsepower,
+                     data = cars, family = Gamma)

> full_summary(model_gamma, cars)
Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 5.970e-05 6.500e-06 9.185 < 2e-16 ***
Weight      -4.608e-09 8.885e-10 -5.186 3.33e-07 ***
Wheelbase    1.598e-07 7.355e-08 2.173 0.0303 *
Horsepower  -1.053e-07 4.451e-09 -23.670 < 2e-16 ***
(Dispersion parameter for Gamma family taken to be 0.09665722)

Null deviance: 106.305 on 427 degrees of freedom
Residual deviance: 36.419 on 424 degrees of freedom
AIC: 8894.6

Number of Fisher Scoring iterations: 7

# A tibble: 1 × 9
logLik  AIC    BIC deviance df.residual      R2      RMSE Value_DF Chi_DF
<dbl> <dbl> <dbl>      <dbl>      <int> <dbl> <dbl> <dbl> <dbl>
1 -4442. 8895. 8915. 36.4 424 0.282 14928. 0.0859 0.0967

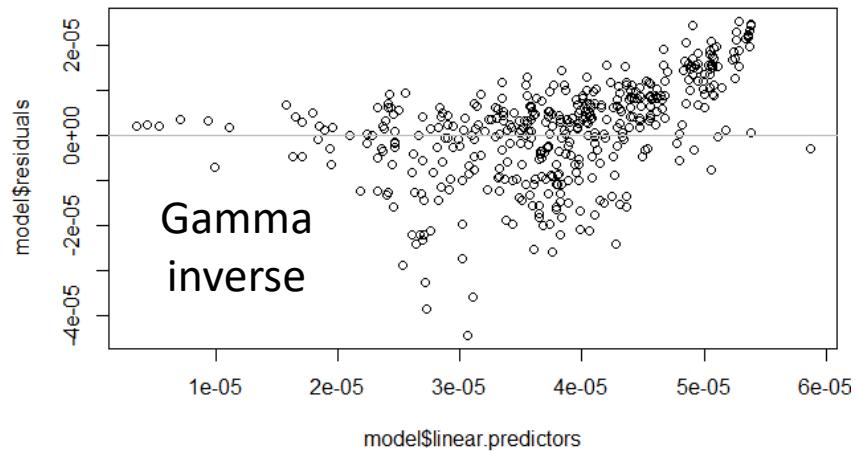
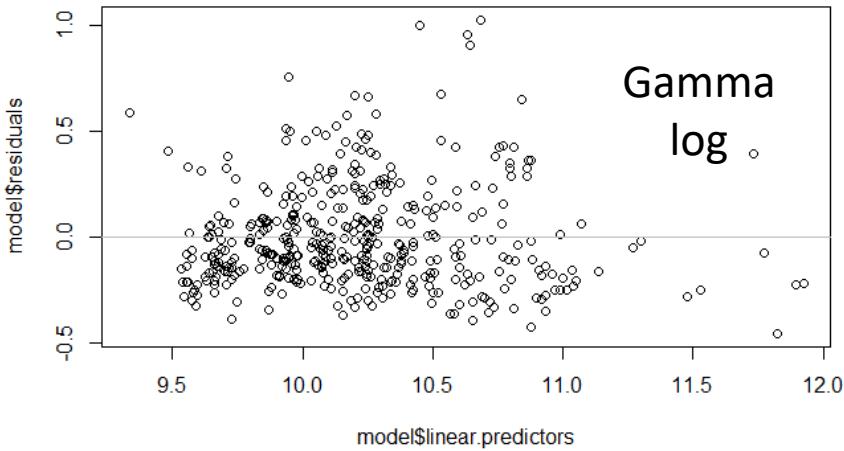
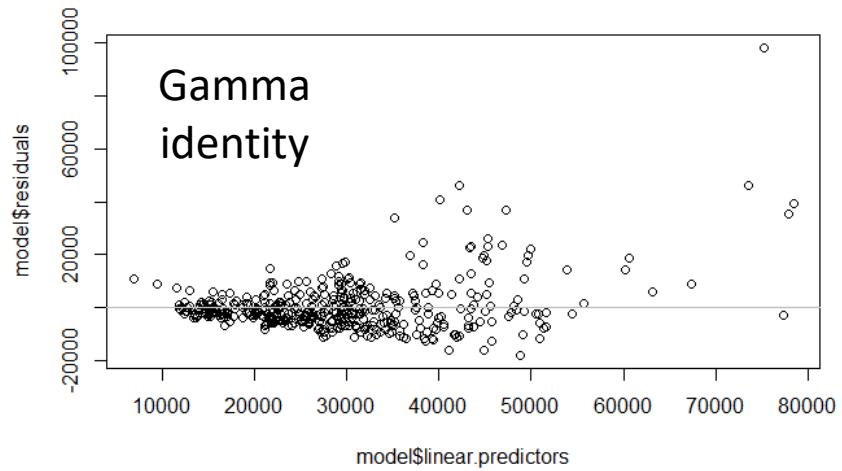
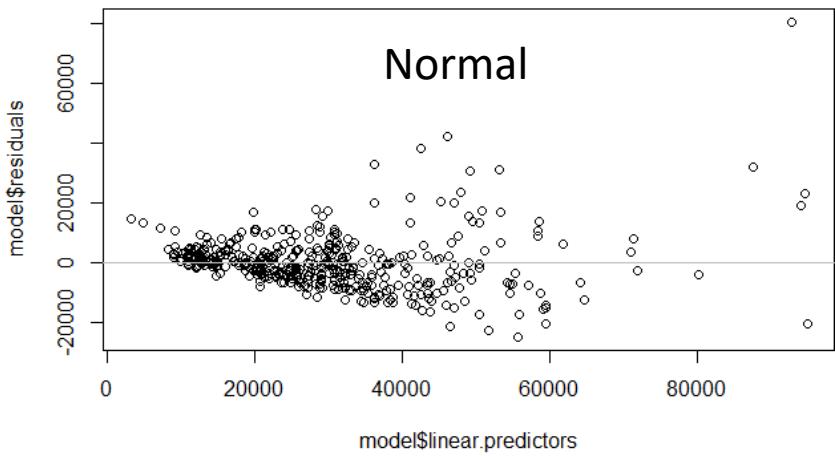
:          p.value      ci.2.5 %      ci.97.5 %
(Intercept) 1.840711e-18 4.677967e-05 7.225646e-05
Weight       3.332815e-07 -6.338232e-09 -2.854686e-09
Wheelbase    3.030142e-02 1.713556e-08 3.055529e-07
Horsepower   1.489901e-79 -1.138807e-07 -9.645211e-08

> plot_predict(model_gamma, cars)
Car Invoice p_Invoice
1   Acura MDX 33337 35432.14
2   Acura RSX Type S 2dr 21761 23820.87
3   Acura TSX 4dr 24647 24669.05
4   Acura TL 4dr 30299 31198.87
5   Acura 3.5 RL 4dr 39014 27392.47
6 Acura 3.5 RL w/Navigation 4dr 41100 27437.49

```

Сравнение адекватности различных моделей

```
> model_normal <- glm(Invoice ~ Weight + Wheelbase + Horsepower, data = cars)
> model_gamma_id <- glm(Invoice ~ Weight + Wheelbase + Horsepower,
+                         data = cars, family = Gamma(link = identity))
> model_gamma_log <- glm(Invoice ~ Weight + Wheelbase + Horsepower,
+                         data = cars, family = Gamma(link = log))
> model_gamma_inv <- glm(Invoice ~ Weight + Wheelbase + Horsepower,
+                         data = cars, family = Gamma(link = inverse))
```



Формальное сравнение моделей

$$MSE = \frac{SSE}{df_E} = \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{n - p}$$

$$R^2 = 1 - \frac{SSE}{SST} = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}$$

$$adj. R^2 = 1 - \frac{SSE / df_E}{SST / df_T} = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2 / (n - p)}{\sum_{i=1}^n (y_i - \bar{y})^2 / (n - 1)}$$

Также часто используется RMSE=SQRT(MSE)

Модель	RMSE	R2
Normal	9402.22	0.6
Gamma inverse	37385.16	0.28
Gamma identity	10021.69	0.26
Gamma log	9476.45	0.7

Пуассоновская регрессия

- Предполагается, что зависимая переменная принадлежит Пуассоновскому распределению
- Используется для моделирования кол-ва наступлений события или доли (rate) наступлений события как функции от набора предикторов
- наиболее подходит для редких событий

- Распределение зависимой переменной должно иметь маленькое среднее (<10 или даже <5, в идеале ~1)
- Иначе гамма и логнормальное распределение может быть лучше, если распределение сильно асимметричное
- или нормальное, если распределение не очень асимметричное

- Модель:

$$\log(\mu) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k$$

Функция
связи

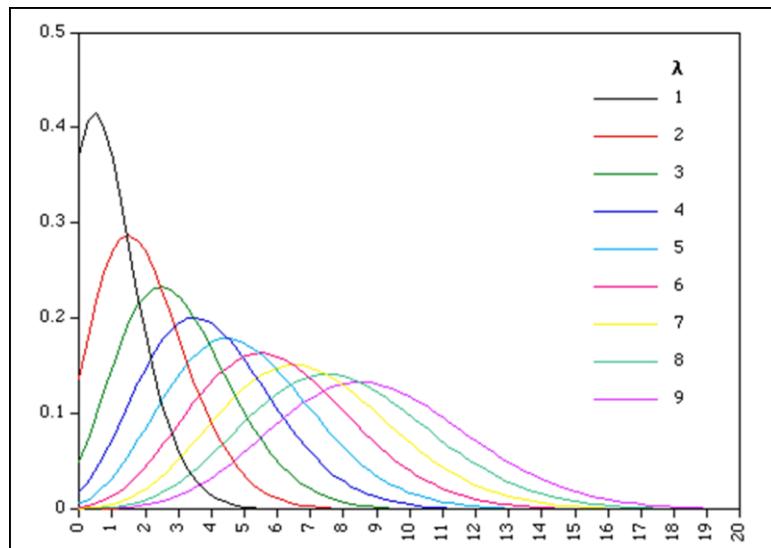
$$\mu = e^{(\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_k X_k)}$$

$$= e^{\beta_0} \cdot e^{\beta_1 X_1} \cdot e^{\beta_2 X_2} \cdots e^{\beta_k X_k}$$

Сравнение нормального и пуассоновского распределения

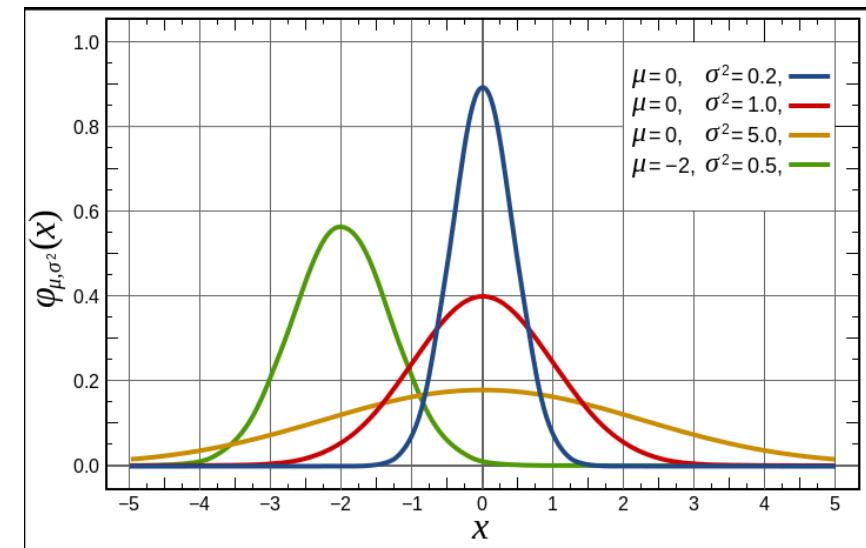
Пуассоновское распределение

- Асимметрично с тяжелым правым хвостом
- Моделирует редкие события
- Положительные значения
- Однопараметрическое распределение (дисперсия равна мат. ожиданию)



Нормальное распределение

- Симметрично
- Любые по знаку значения
- Два независимых параметра



С ростом мат. ожидания пуассоновское распределение становится все симметричней и приближается по свойствам к нормальному, отсюда следует область приложения - **редкие события**

Интерпретация параметров пуассоновской регрессии

$e^{\hat{\beta}} =$ мультипликативный эффект на $\hat{\mu}$
от изменения на одну единицу по X.

Пример 1, если

$e^{\hat{\beta}_1} =$ 1.20, увеличение X_1 на одну единицу вызывает
20% увеличение оцениваемого среднего.

Пример 2, если

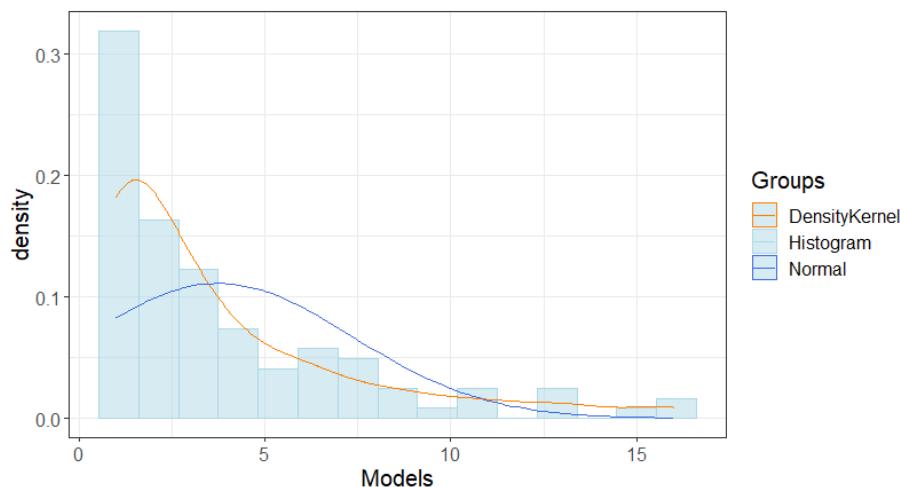
$e^{\hat{\beta}_2} =$ 0.80, увеличение X_2 на одну единицу вызывает
20% уменьшение оцениваемого среднего

Естественным образом похоже на отношение шансов в
логистической регрессии

Пример пуассоновской регрессии

- Модельная задача:
 - Посчитаем число различных моделей в разрезе производитель, страна, тип кузова и найдем как это число зависит от агрегированных значений остальных числовых предикторов

```
> VARS <- c("Invoice", "MPG_Highway", "MPG_City",
+           "Horsepower", "EngineSize", "Weight")
> v <- c()
> for (var in VARS){
+   for (agg in c("min", "max", "avg")){
+     v <- c(v, paste(agg, "(", var, ") as ", agg, var, sep = ""))
+   }
+ }
> v <- paste(c(v, ""), collapse = ", ")
> query <- paste(c("select", v,
+                   "count(*) as Models, Make, Type, Origin",
+                   "from cars group by Make, Type, Origin"),
+                   collapse = " "))
> cars_counts <- sqldf(query)
```



Пример пуассоновской регрессии

```
> model_poiss <- glm(Models ~ Origin + avgWeight + maxInvoice + minHorsepower,  
+                      data = cars_counts, family = poisson(link = "log"))  
>  
> full_summary(model_poiss, cars_counts)
```

```
Coefficients:  
            Estimate Std. Error z value Pr(>|z|)  
(Intercept) 1.966e+00 2.663e-01 7.380 1.58e-13 ***  
OriginEurope -3.128e-02 1.629e-01 -0.192 0.848  
OriginUSA     2.073e-01 1.226e-01 1.690 0.091 .  
avgWeight    1.350e-04 9.404e-05 1.436 0.151  
maxInvoice   1.509e-05 2.062e-06 7.318 2.51e-13 ***  
minHorsepower -1.005e-02 1.234e-03 -8.141 3.94e-16 ***
```

```
Null deviance: 315.95 on 113 degrees of freedom  
Residual deviance: 190.27 on 108 degrees of freedom  
AIC: 529.17
```

```
Number of Fisher Scoring iterations: 5
```

```
# A tibble: 1 × 9  
  logLik  AIC   BIC deviance df.residual      R2    RMSE Value_DF Chi_DF  
  <dbl> <dbl> <dbl>      <dbl>     <int> <dbl> <dbl> <dbl> <dbl>  
1  -259.   529.  546.      190.       108  0.345  2.89  1.76  1.90
```

	estimate	std.error	statistic	p.value	ci.2.5 %	ci.97.5 %
(Intercept)	1.965538e+00	2.663181e-01	7.3804134	1.577988e-13	1.444393e+00	2.488503e+00
OriginEurope	-3.127737e-02	1.628919e-01	-0.1920131	8.477320e-01	-3.552290e-01	2.839056e-01
OriginUSA	2.072741e-01	1.226418e-01	1.6900774	9.101315e-02	-3.361973e-02	4.475207e-01
avgWeight	1.349990e-04	9.403491e-05	1.4356262	1.511087e-01	-4.971053e-05	3.190290e-04
maxInvoice	1.509209e-05	2.062201e-06	7.3184398	2.508705e-13	1.098745e-05	1.908716e-05
minHorsepower	-1.004917e-02	1.234460e-03	-8.1405409	3.935159e-16	-1.251872e-02	-7.682301e-03

Должны быть близки к 1 для адекватной модели

Чрезмерная дисперсия

- Модели пуассоновской регрессии предполагают равенство среднего и дисперсии.
- При моделировании кол-ва (count data) часто дисперсия превосходит среднее.
- Недооценены стандартные ошибки оценок параметров.
- Переоценены статистики тестов и p -values.

- Причины:
 - Модель недоопределена – отсутствуют важные предикторы
 - Выбросы
 - Корреляция

- Что делать?
 - Проверить данные на случайность, артефакты, выбросы и попробовать разные комбинации предикторов
 - Если не помогло, то переходить к другому распределению отклика – отрицательному биномиальному

Отрицательное биномиальное распределение

- Сл. величина Y распределена $NB(k, p)$ – $Y=n$ показывает номер k -го успеха в серии из n независимых испытаний Бернули с параметром p , или число испытаний для достижения заданного числа успехов
- для количественных данных допускает превышение дисперсии над средним
- дает большую гибкость (*чем пуассоновское распр.*) в моделировании связи между средним и дисперсией зависимой переменной

Отклик	Распр.	Функция связи	Дисперсия
счетчик	Neg. Binomial	Log	$\mu + k\mu^2$

- Параметр дисперсии k - постоянный по всем наблюдениям.
- $k = 0$ – пуассоновская регрессия. $k > 0$ – случай чрезмерной дисперсии, стандартные ошибки увеличиваются. Полученные по модели значения Y и параметров будут похожи, но дисперсия правильно учтена = корректные p-value для тестов.

Пример отрицательной биномиальной регрессии

```
> model_nb <- glm.nb(Models ~ Origin + avgWeight + maxInvoice + minHorsepower,  
+ data = cars_counts, link = log)
```

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.966e+00	3.573e-01	5.501	3.77e-08 ***
OriginEurope	-1.121e-01	2.224e-01	-0.504	0.614
OriginUSA	2.117e-01	1.666e-01	1.270	0.204
avgWeight	5.756e-05	1.240e-04	0.464	0.642
maxInvoice	1.942e-05	3.295e-06	5.895	3.75e-09 ***
minHorsepower	-9.315e-03	1.540e-03	-6.048	1.47e-09 ***

Null deviance: 167.690 on 113 degrees of freedom
Residual deviance: 97.532 on 108 degrees of freedom

	ci.2.5 %	ci.97.5 %
(Intercept)	1.236467e+00	2.705267e+00
OriginEurope	-5.543744e-01	3.251607e-01
OriginUSA	-1.119347e-01	5.360867e-01
avgWeight	-1.981930e-04	3.148385e-04
maxInvoice	1.170101e-05	2.776267e-05
minHorsepower	-1.236175e-02	-6.432337e-03

negbin

Coefficients:

	Estimate	Std. Error	z value	Pr(> z)
(Intercept)	1.966e+00	2.663e-01	7.380	1.58e-13 ***
OriginEurope	-3.128e-02	1.629e-01	-0.192	0.848
OriginUSA	2.073e-01	1.226e-01	1.690	0.091 .
avgWeight	1.350e-04	9.404e-05	1.436	0.151
maxInvoice	1.509e-05	2.062e-06	7.318	2.51e-13 ***
minHorsepower	-1.005e-02	1.234e-03	-8.141	3.94e-16 ***

Null deviance: 315.95 on 113 degrees of freedom
Residual deviance: 190.27 on 108 degrees of freedom

	ci.2.5 %	ci.97.5 %
(Intercept)	1.444393e+00	2.488503e+00
OriginEurope	-3.552290e-01	2.839056e-01
OriginUSA	-3.361973e-02	4.475207e-01
avgWeight	-4.971053e-05	3.190290e-04
maxInvoice	1.098745e-05	1.908716e-05
minHorsepower	-1.251872e-02	-7.682301e-03

poisson

- Все статистики стали лучше
- Увеличилась ошибка в оценках коэф.
- Увеличились доверительные интервалы

negbin

logLik	AIC	BIC	deviance	df.residual	R2	RMSE	Value_DF	Chi_DF
<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
1 -242.5696	499.	518.	97.5	108	0.101	3.39	0.903	0.962

logLik	AIC	BIC	deviance	df.residual	R2	RMSE	Value_DF	Chi_DF
<dbl>	<dbl>	<dbl>	<dbl>	<int>	<dbl>	<dbl>	<dbl>	<dbl>
1 -259.	529.	546.	190.	108	0.345	2.89	1.76	1.90

poisson

glm.nb(formula, data, weights, subset, ..., contrasts = NULL, link = log)

Нелинейные зависимости

1. Нелинейная регрессия

- Необходимо задать уравнение регрессии (отбора признаков нет)
- Результат зависит от метода оптимизации и начального приближения

2. Полиномиальная регрессия

- Задать в явном виде или указать порядок полного полинома (экспоненциальный рост числа членов полинома)
- возможно с автоматическим отбором членов полинома пошаговыми методами

3. Непараметрические регрессии (сплайны и локальные взвешенные регрессии)

- Требуется хранение как минимум части тренировочного набора (иногда большой и даже всей)
- Не устойчивы к проклятию размерности

Задача оптимизации для нелинейных регрессий

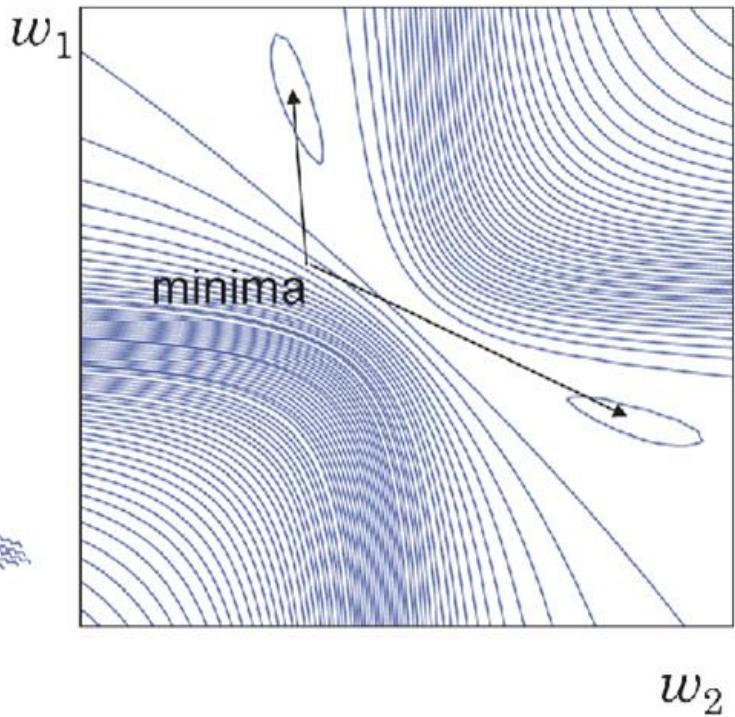
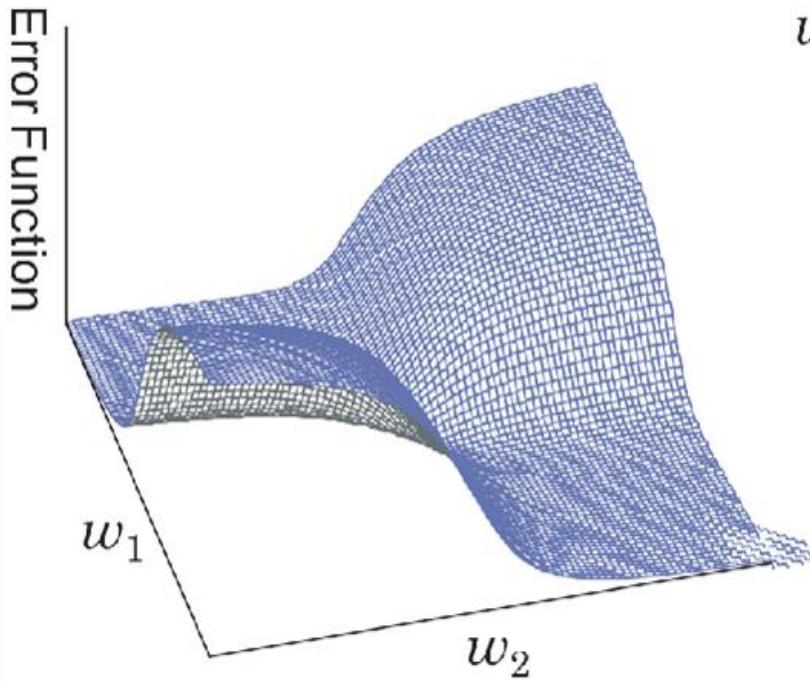
- Требует определения уравнения и начального приближения в явном виде
- Можно использовать бутстреппинг для оценки доверительных интервалов параметров
- Нужно строить диагностические графики и считать статистики
- Требуется выбирать один из методов оптимизации
- Методы первого порядка – градиентные (используют шаг «вдоль» направления градиента – вектора первых производных)
 - выбор шага (константа, дробный выбор, адаптивный, наискорейший)
 - выбор направления (с учетом предыдущих шагов, например сопряженные градиенты)

$$y = f(\mathbf{x} + \Delta\mathbf{x}) \approx f(\mathbf{x}) + J(\mathbf{x})\Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}^T H(\mathbf{x})\Delta\mathbf{x}$$

- Методы второго порядка – ньютоновские (используют матрицу вторых производных Гессе для «выбора шага»)
 - проблема – вычисление обратной матрицы Гессе на каждом шаге

Итерационные методы

$$\mathbf{w}^{(t+1)} = \mathbf{w}^{(t)} + \boldsymbol{\delta}^{(t)}$$



Градиентный:

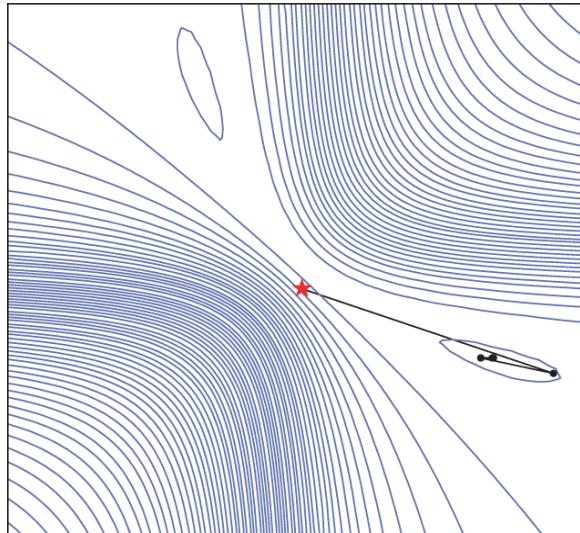
$$\boldsymbol{\delta}^{(t)} = -\eta \nabla g^{(t)}$$

Ньютона:

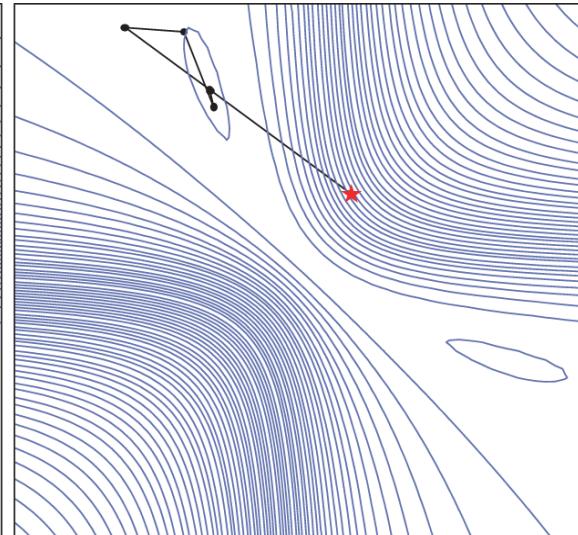
$$\boldsymbol{\delta}^{(t)} = -[\mathbf{H}^{(t)}]^{-1} \nabla g^{(t)}$$

Левенберга — Марквардта

$$\delta^{(t)} = -(\mathbf{J}^{(t) \top} \mathbf{J}^{(t)} + \lambda^{(t)} \mathbf{I})^{-1} \mathbf{J}^{(t) \top} \mathbf{r}^{(t)}$$



8 iterations

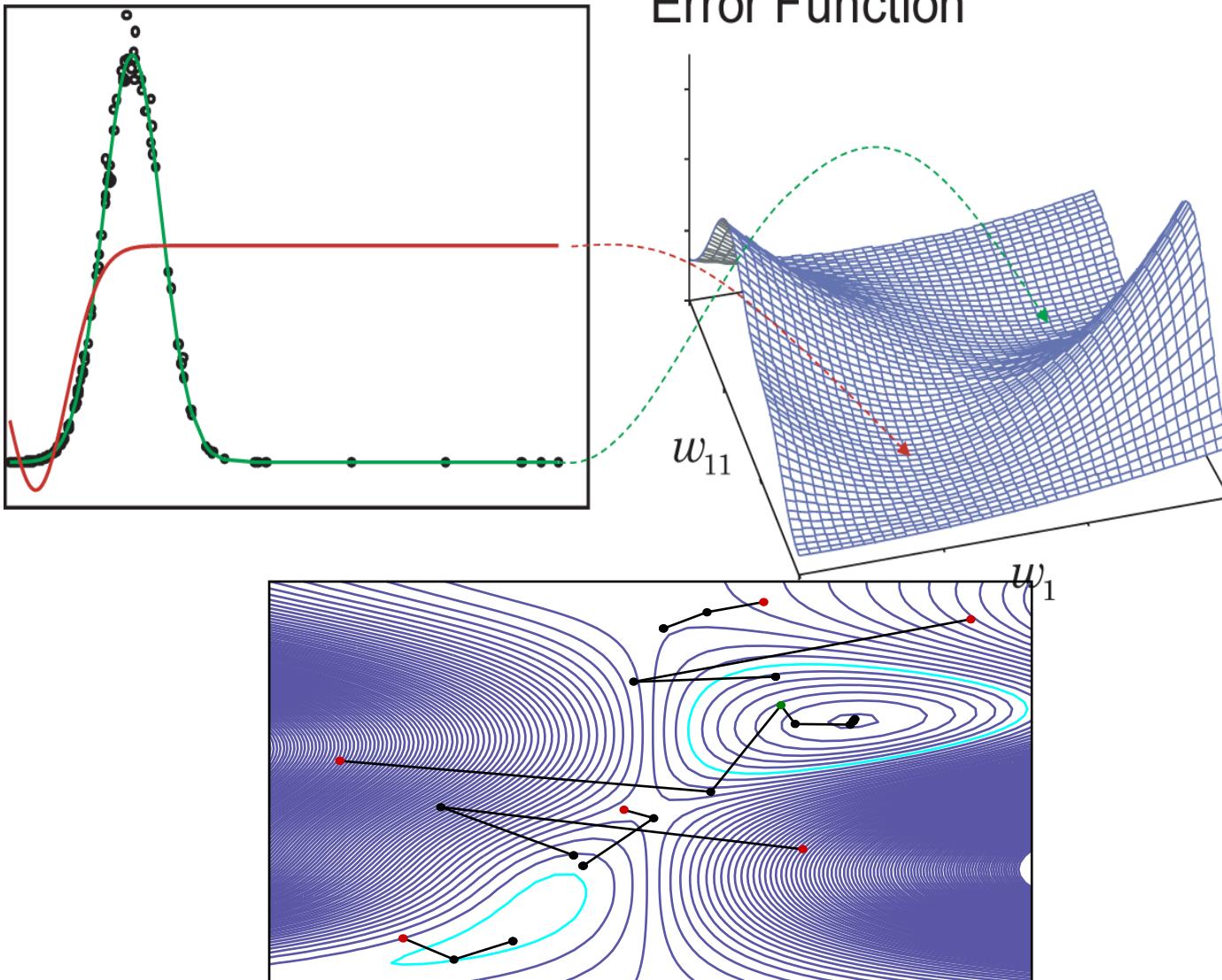


6 iterations

Комбинация градиентного (лямбда велико) и Ньютона (лямбда=0),
Применим для небольшого количества переменных <100

$$\mathbf{J} = \begin{bmatrix} \frac{\partial \mathbf{f}}{\partial x_1} & \cdots & \frac{\partial \mathbf{f}}{\partial x_n} \end{bmatrix} = \begin{bmatrix} \nabla^T f_1 \\ \vdots \\ \nabla^T f_m \end{bmatrix} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

Проблема начального приближения, сходимости и локальных минимумов



Нелинейная регрессия наименьших квадратов NLS

```
nls(formula, data, start, control, algorithm, trace, subset, weights,  
na.action, model, lower, upper, ...)
```

```
> full_formula<-formula(Invoice ~ exp(b0 + b1*tanh(a1+all*Weight+a12*Horsepower))  
+ + b2*tanh(a2+a21*Weight+a22*Horsepower))  
> nlc <- nls.control(maxiter = 500, warnOnly = TRUE, tol=0.1)  
> model_nls <- nls(full_formula, control=nlc, data = cars_std, algorithm="port", trace=TRUE)  
0: 7166.6338: 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000 1.00000  
1: 2172.0752: 0.751796 0.799671 0.763639 1.01488 1.03558 -0.309135 0.344117 1.17494 1.24251  
2: 793.83955: 0.430682 0.618204 0.394436 1.04974 1.12243 -0.234738 0.959517 1.14461 1.13279  
3: 555.49382: 0.203014 0.125162 0.00370974 0.172020 1.16223 0.145369 2.50137 -6.85691 -4.19512  
4: 211.83884: -0.337512 0.641323 -0.200122 -1.09542 1.05364 -0.348917 1.87378 -12.0354 -6.40515  
5: 171.81325: -0.431154 1.14166 -0.441520 0.120241 1.10244 -0.339340 2.29544 -10.1874 -5.54393  
6: 131.83533: -0.699655 1.67642 -0.580530 -0.517399 0.272756 -0.486423 1.94069 -3.86421 -3.81641  
7: 88.217521: -0.699916 1.74764 -0.583027 -0.432482 0.595444 -0.524580 1.97210 -2.96689 -3.65442  
8: 72.016533: -0.699837 2.20184 -0.623585 -0.142073 0.618759 -0.503937 1.72020 -1.72421 -2.64907  
9: 70.190418: -0.696555 2.42004 -0.599684 -0.109857 0.535660 -0.578835 0.504944 0.181206 -0.842951  
117: 55.674383: 0.414611 0.812630 -16.7947 -2.02632 5.37126 -2.13610 0.376023 0.0143497 -0.411181  
118: 55.674383: 0.414609 0.812628 -16.7954 -2.02607 5.37139 -2.13610 0.376023 0.0143501 -0.411182  
119: 55.674383: 0.414609 0.812628 -16.7955 -2.02630 5.37149 -2.13610 0.376023 0.0143501 -0.411182  
120: 55.674383: 0.414609 0.812628 -16.7957 -2.02621 5.37150 -2.13610 0.376023 0.0143501 -0.411182  
Warning message:  
In nls(full_formula, control = nlc, data = cars_std, algorithm = "port", :  
  No starting values specified for some parameters.  
Initializing 'b0', 'b1', 'a1', 'all', 'a12', 'b2', 'a2', 'a21', 'a22' to '1'.  
Consider specifying 'start' or using a selfStart model
```

Нелинейная регрессия с NLS

```
> summary(model_nls)
```

```
Formula: Invoice ~ exp(b0 + b1 * tanh(a1 + a11 * Weight + a12 * Horsepower)) +
    b2 * tanh(a2 + a21 * Weight + a22 * Horsepower)
```

Parameters:

	Estimate	Std. Error	t value	Pr(> t)
b0	0.41461	0.25664	1.616	0.10695
b1	0.81263	0.30149	2.695	0.00731 **
a1	-16.79567	38.04382	-0.441	0.65909
a11	-2.02621	14.23181	-0.142	0.88685
a12	5.37150	14.16092	0.379	0.70464
b2	-2.13610	0.68919	-3.099	0.00207 **
a2	0.37602	0.11467	3.279	0.00113 **
a21	0.01435	0.01935	0.742	0.45865
a22	-0.41118	0.14460	-2.843	0.00468 **

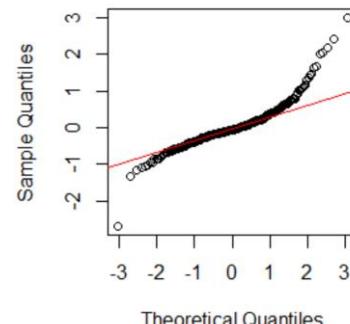
```
Signif. codes: 0 '****' 0.001 '***' 0.01 '**' 0.05 '.' 0.1 ' ' 1
```

```
Residual standard error: 0.5155 on 419 degrees of freedom
```

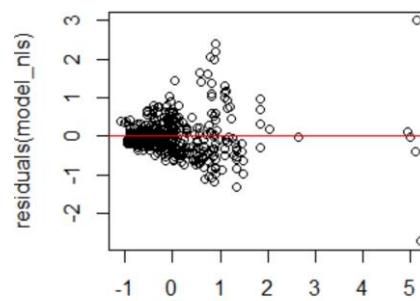
```
Algorithm "port", convergence message: relative convergence (4)
```

```
> qqnorm(residuals(model_nls))
> qqline(residuals(model_nls), col="red")
```

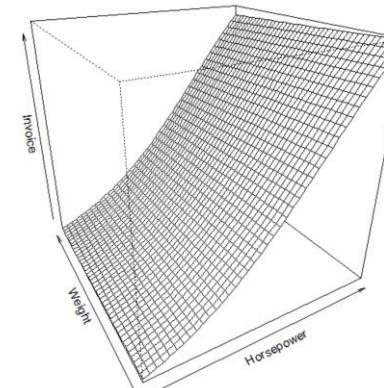
Normal Q-Q Plot



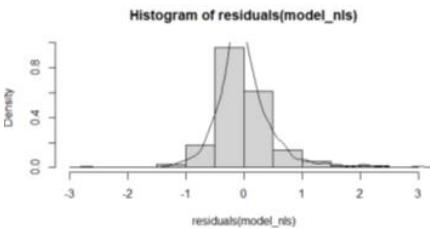
```
> plot(x=predict(model_nls,cars_std),y=residuals(model_nls))
> abline(h=0,col="red")
```



```
> persp(outer(seq(-2,2,0.1),seq(-2,2,0.1),
+ FUN=function(h,w){
+ return(predict(model_nls,list(Horsepower=h,Weight=w)))
+ }), theta=-30,xlab="Horsepower",
+ ylab="Weight",zlab="Invoice", phi=25)
```



```
> hist(residuals(model_nls),prob=TRUE)
> lines(density(residuals(model_nls)))
> boxplot(residuals(model_nls),horizontal=TRUE)
```



Полиномиальная регрессия

Можно задавать предикторы через :, ^, * и I()

```
> poly_model<-lm(Invoice~I(Weight^2)+I(Cylinders^2)+I(Length^2)+  
+ Weight+Cylinders+Length+(Weight+Cylinders+Length)^2,cars)
```

а можно их «предпросчитать» в явном виде:

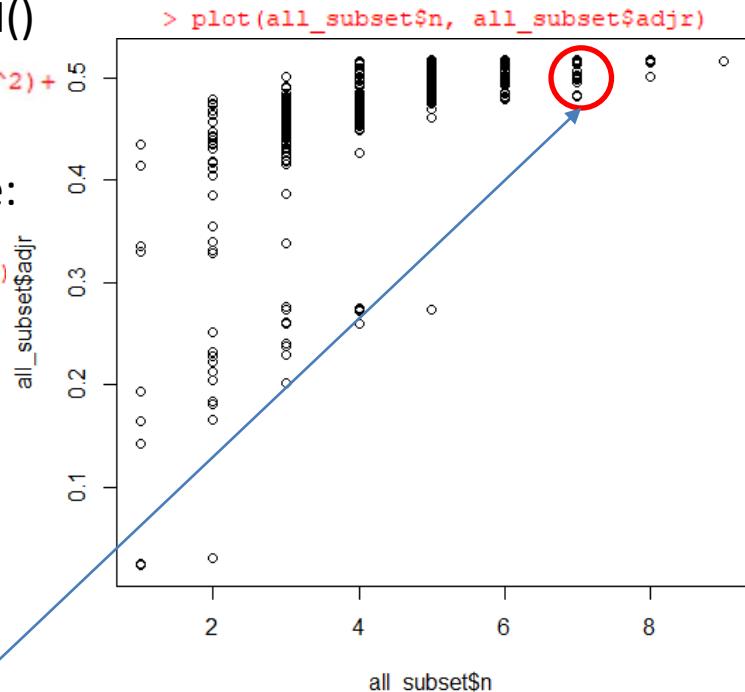
```
> feat_cars4 <- c("Weight", "Length", "Cylinders", "Invoice")  
> cars4 <- data.frame(  
+   Weight2 = cars$Weight^2,  
+   Length2 = cars$Length^2,  
+   Cylinders2 = cars$Cylinders^2,  
+   Cylinders_Length = cars$Cylinders*cars$Length,  
+   Cylinders_Weight = cars$Cylinders*cars$Weight,  
+   Weight_Length = cars$Weight*cars$Length  
+ )  
> cars4[, feat_cars4] <- cars[, feat_cars4]
```



```
> model <- lm(Invoice ~ ., data = cars4)  
> all_subset <- ols_step_all_possible(model)
```

```
> head(all_subset[order(all_subset$adjr, decreasing = TRUE),])
```

Index	N	Predictors	R-Square	Adj. R-Square
474	466	7 Weight2 Length2 Cylinders2 Cylinders_Length Weight_Length Length Cylinders	0.5265030	0.5185737
297	256	5 weightz cylinderez cylinders_lengthn weight_lengthn cylinderns	0.5239425	0.5182752
299	257	5 Weight2 Cylinders2 Cylinders_Length Weight Cylinders	0.5238458	0.5181773
424	382	6 Weight2 Cylinders2 Cylinders_Length Weight_Length Weight Cylinders	0.5249634	0.5181609
388	383	6 Weight2 Length2 Cylinders2 Cylinders_Length Weight_Length Cylinders	0.5245119	0.5177030
421	384	6 Weight2 Cylinders2 Cylinders_Length Cylinders_Weight Weight Cylinders	0.5243967	0.5175861



Сплайны

Кубические сплайны с узлами ξ_k , $k = 1, \dots, K$ представляют собой кусочно-кубический многочлен с непрерывными производными до второго порядка в каждом узле.

Мы можем представить эту модель со степенными базисными функциями

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \beta_2 b_2(x_i) + \cdots + \beta_{K+3} b_{K+3}(x_i) + \epsilon_i,$$

$$\begin{aligned} b_1(x_i) &= x_i \\ b_2(x_i) &= x_i^2 \\ b_3(x_i) &= x_i^3 \\ b_{k+3}(x_i) &= (x_i - \xi_k)_+^3, \quad k = 1, \dots, K \end{aligned}$$

где

$$(x_i - \xi_k)_+^3 = \begin{cases} (x_i - \xi_k)^3 & \text{if } x_i > \xi_k \\ 0 & \text{otherwise} \end{cases}$$

Сглаживание сплайнами

Подгонка гладкой функцией $g(x)$:

$$\underset{g \in \mathcal{S}}{\text{minimize}} \sum_{i=1}^n (y_i - g(x_i))^2 + \lambda \int g''(t)^2 dt$$

- Первое слагаемое - RSS и он нацелен на то, чтобы $g(x)$ соответствовала данным в каждом x_i .
- Второе слагаемое - это штраф за *грубое приближение* и он управляет тем, насколько $g(x)$ «извилистая». Он варьируется *параметром настройки* $\lambda \geq 0$.
 - Чем меньше, тем более извилистая функция, в конечном счете интерполирующая y_i , когда $\lambda = 0$.
 - Когда $\lambda \rightarrow \infty$, функция $g(x)$ становится линейной.

```
Tps(x, Y, m = NULL, p = NULL, scale.type = "range", lon.lat = FALSE,  
      miles = TRUE, method = "GCV", GCV = TRUE, ...)  
fastTps(x, Y, m = NULL, p = NULL, aRange, lon.lat = FALSE,  
      find.trA = FALSE, REML = FALSE, theta=NULL, ...)
```

Сглаживание сплайнами

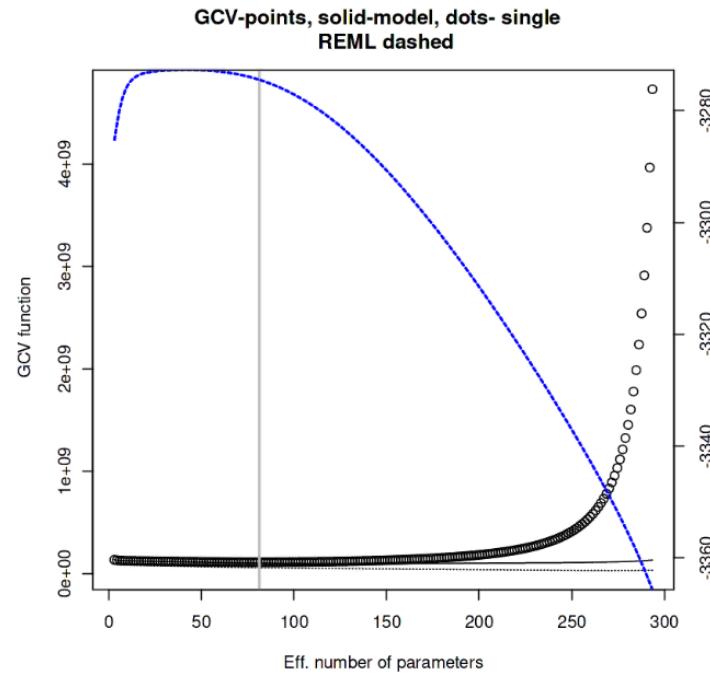
- Автоматический выбор (если не задано) параметра регуляризации с помощью кросс-валидации (GCV)
- Есть полу-параметрические модели (часть регрессия, часть сплайны)
- Диагностические графики и статистики

```
> model_tps <- Tps(x = as.matrix(cars[, c("Horsepower", "Length")]),
+                      Y = cars$Invoice, method = "GCV")

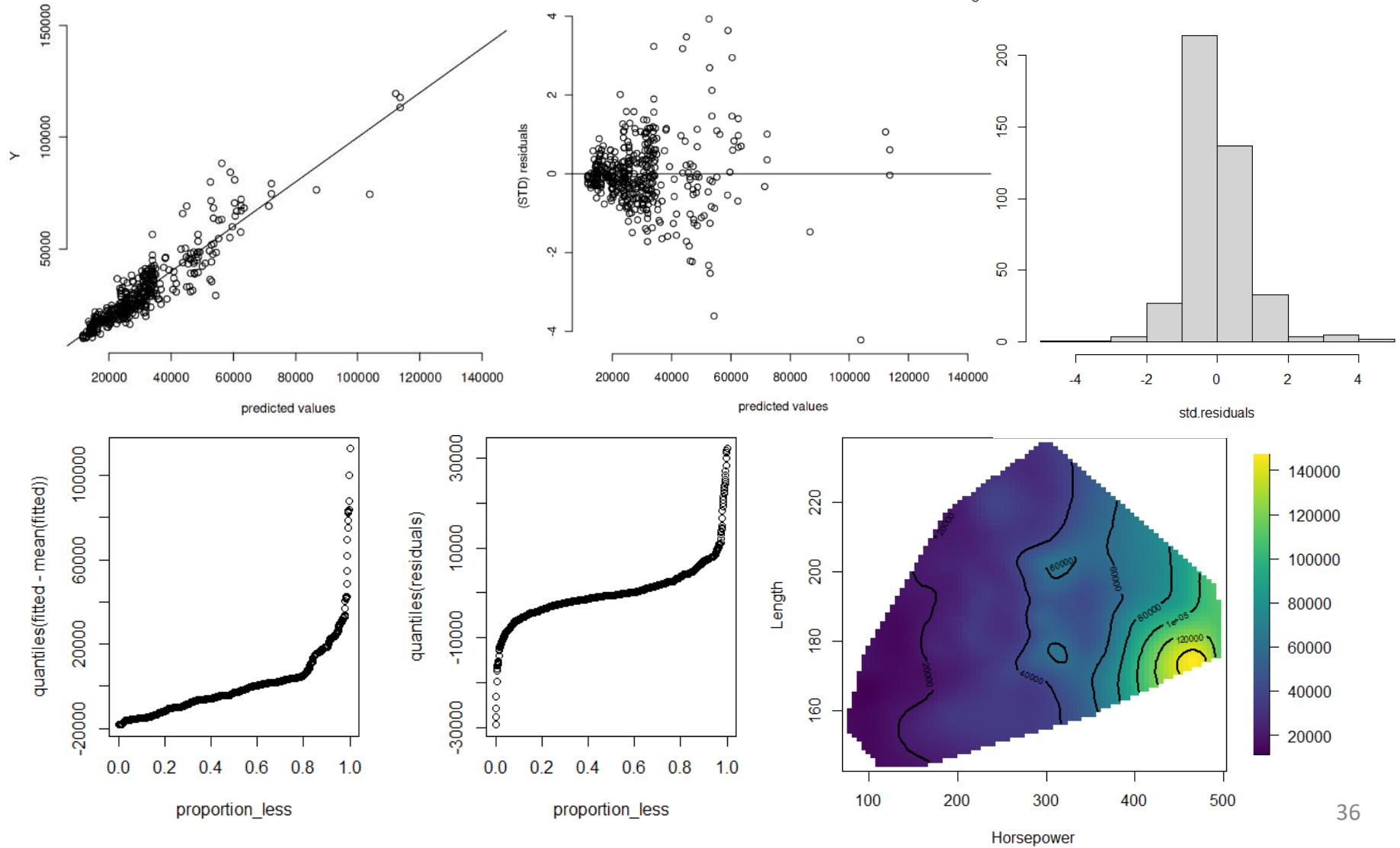
Number of Observations:          428
Number of unique points:        309
Number of parameters in the null space 3
Parameters for fixed spatial drift 3
Effective degrees of freedom:   81.4
Residual degrees of freedom:    346.6
MLE tau                         6875
GCV tau                          6971
Pure error tau                   3395
MLE sigma                        1.55e+11
Scale passed for covariance (sigma) <NA>
Scale passed for nugget (tau^2)    <NA>
Smoothing parameter lambda      0.000305

DETAILS ON SMOOTHING PARAMETER:
Method used: GCV Cost: 1
lambda      trA      GCV  GCV.one GCV.model     tauHat
3.050e-04 8.143e+01 1.135e+08 6.000e+07 1.038e+08 6.971e+03

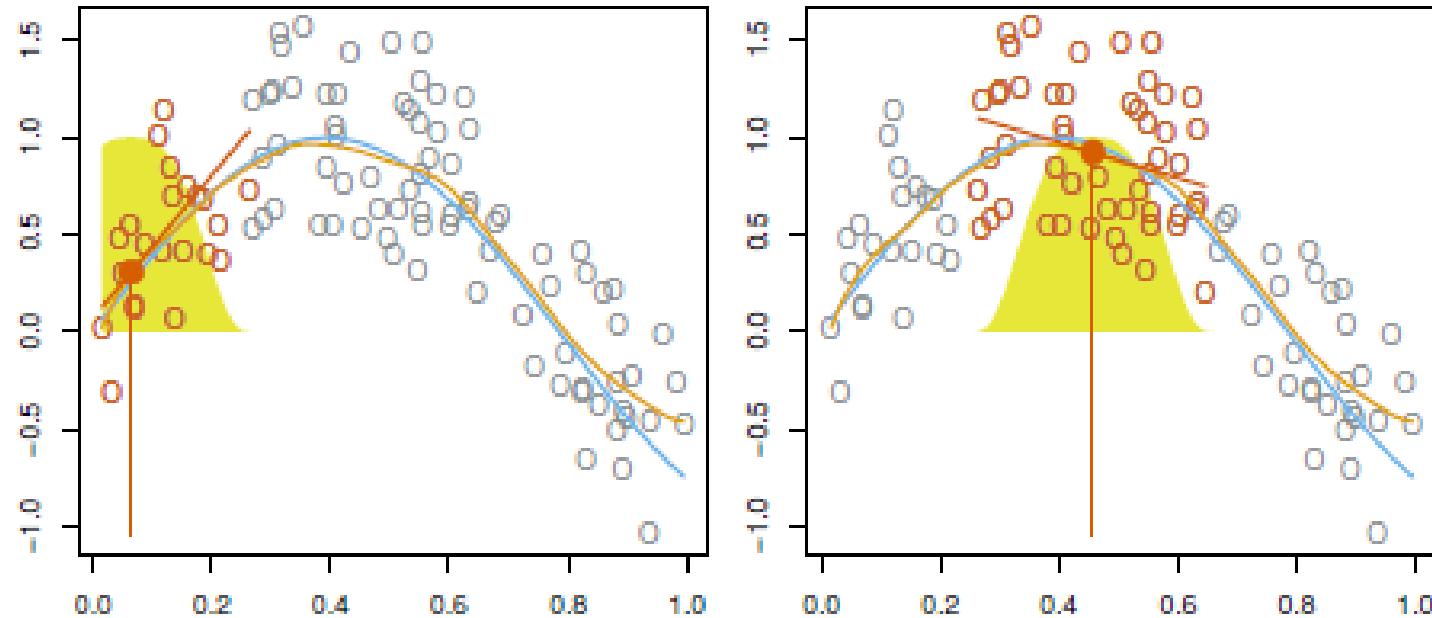
Summary of all estimates found for lambda
      lambda      trA      GCV tauHat -lnLike Prof converge
GCV      3.050e-04 81.43 1.135e+08 6971      3274     20
GCV.model 7.591e-05 131.65 9.965e+07 5907       NA     19
GCV.one   2.401e-06 270.95 3.222e+07 3438       NA     19
RMSE        NA       NA      NA       NA       NA       NA
pure error 2.027e-06 275.16 1.067e+09 3395      3352     NA
REML      1.595e-03 42.22 1.185e+08 8068      3273      2
```



Сглаживание сплайнами



Локальная взвешенная регрессия



- С помощью скользящей весовой функции отдельно подгоняем линейные участки по диапазону X с помощью взвешенных наименьших квадратов
- Параметр регуляризации подбирается кросс-валидацией или через информационные критерии

Локальная регрессия на решетке

```

> ctrl <- trainControl(method = "cv", number = 7)
> grid <- expand.grid(span = seq(0.1, 0.9, len = 20), degree = 1)
>
> loess_model <- train(Invoice ~ Length + Horsepower, data = cars,
+                         method = "gamLoess", tuneGrid=grid, trControl = ctrl)
>
> print(loess_model)

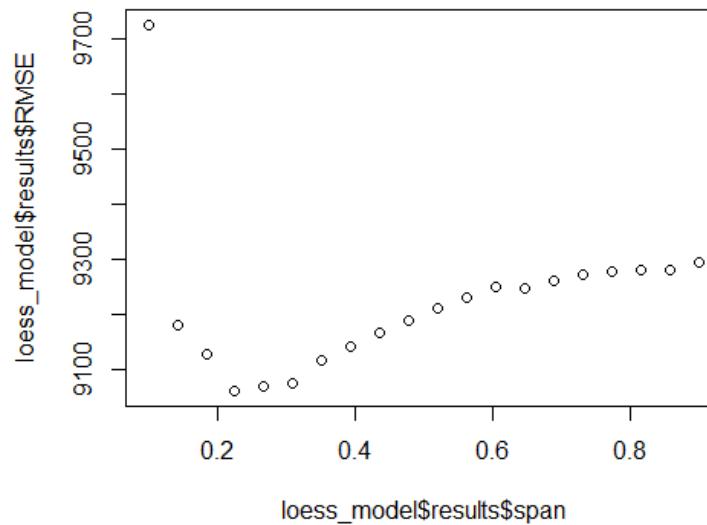
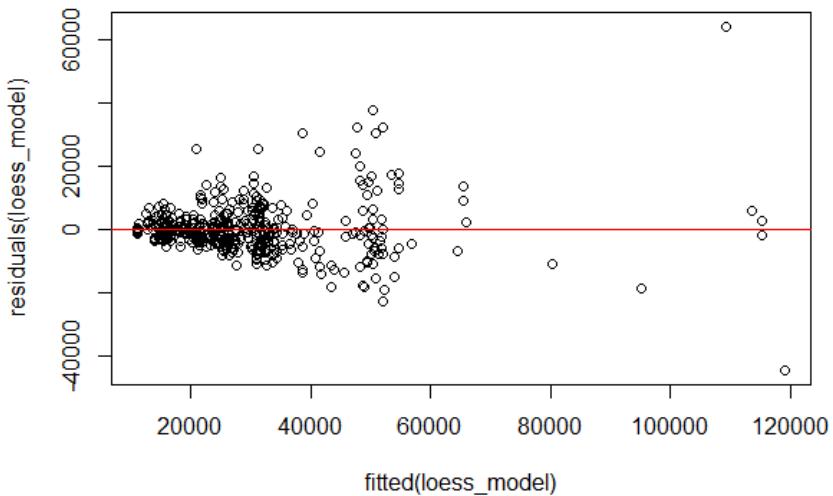
Null Deviance: 132901324092 on 427 degrees of freedom
Residual Deviance: 30967546072 on 409.0014 degrees of freedom
AIC: 9000.16

Number of Local Scoring Iterations: NA

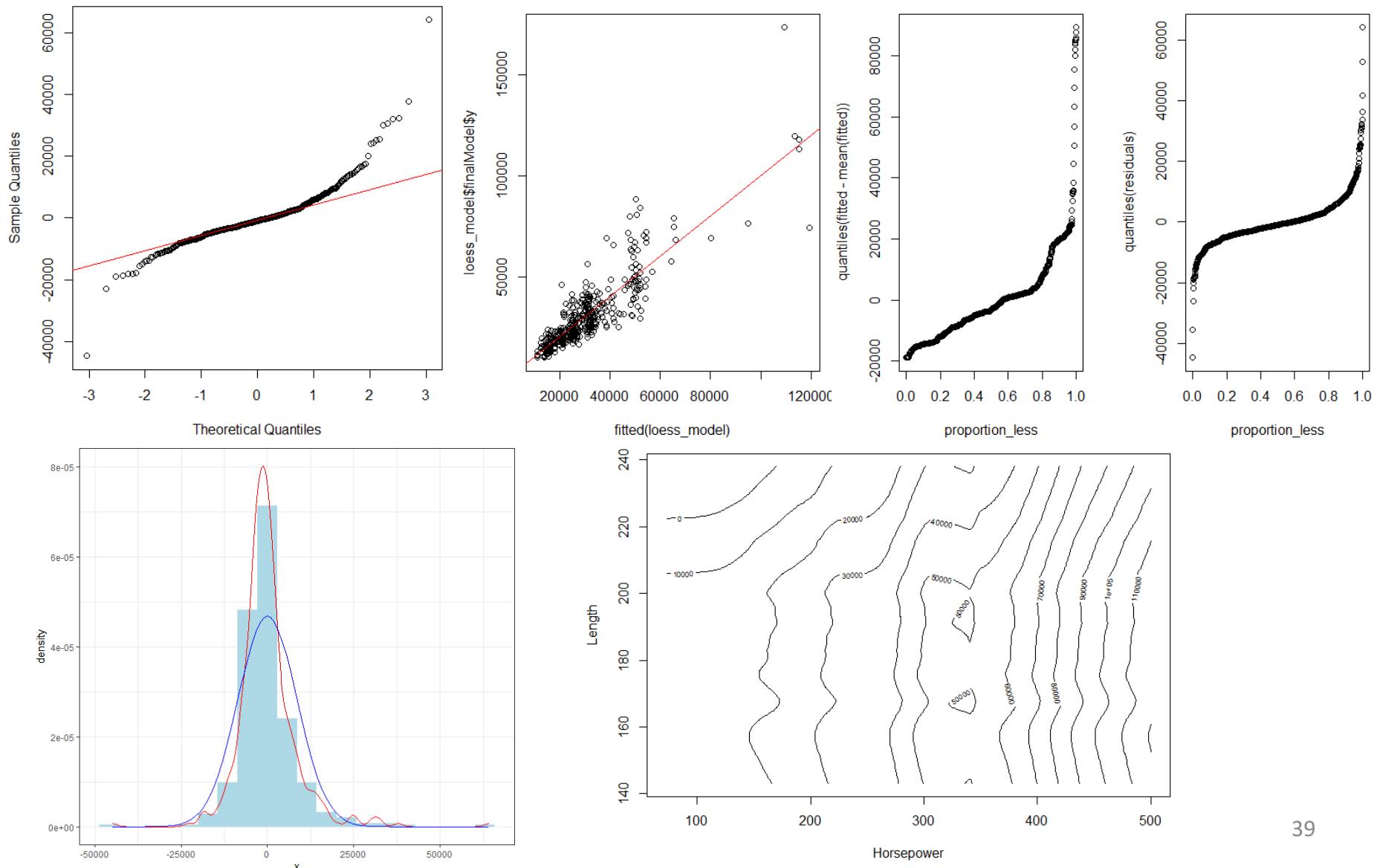
Anova for Parametric Effects
                                         Df      Sum Sq   Mean Sq F value    Pr(>F)
lo(Length, span = 0.226315789473684, degree = 1)  1 6.3462e+09 6.3462e+09  83.817 < 2.2e-16 ***
lo(Horsepower, span = 0.226315789473684, degree = 1) 1 8.3299e+10 8.3299e+10 1100.161 < 2.2e-16 ***
Residuals                                         409 3.0968e+10 7.5715e+07
                                         0.5210526 9210.656 0.7323683 6057.019
                                         0.5631579 9229.311 0.7310900 6077.240
                                         0.6052632 9248.645 0.7301589 6093.991
                                         0.6473684 9247.299 0.7304381 6092.797
                                         0.6894737 9259.082 0.7297023 6105.199
                                         0.7315789 9270.536 0.7289893 6118.137
                                         0.7736842 9276.457 0.7286209 6127.327
                                         0.8157895 9279.548 0.7286104 6139.807
                                         0.8578947 9280.035 0.7287368 6145.170
                                         0.9000000 9294.159 0.7284041 6154.170

Anova for Nonparametric Effects
                                         Npar Df   Npar F    Pr(F)
(Intercept)
lo(Length, span = 0.226315789473684, degree = 1)     8  3.0644 0.002362 **
lo(Horsepower, span = 0.226315789473684, degree = 1)    8 12.1336 1.11e-15 ***

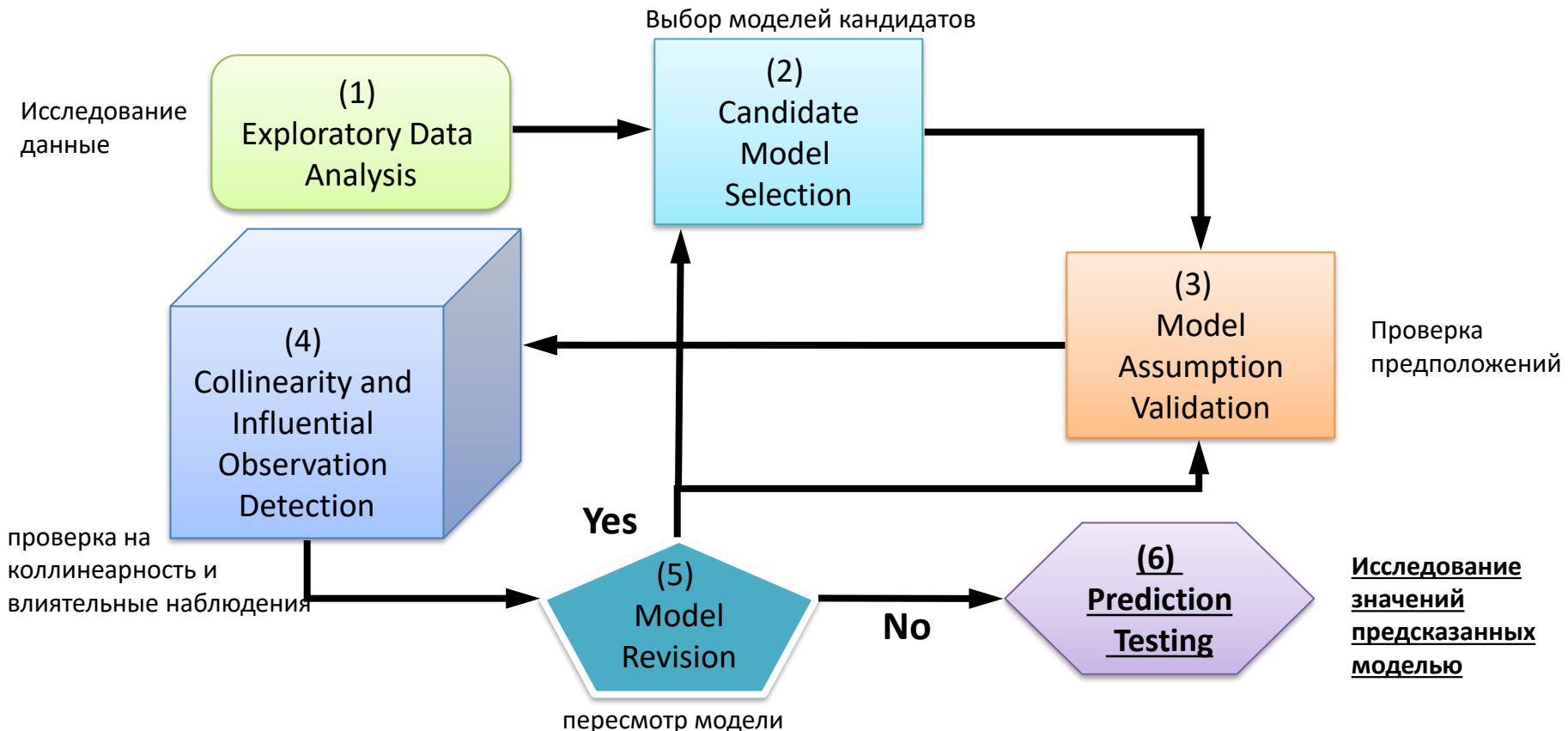
```



Локальная регрессия



Процесс разработки модели



Сложность модели

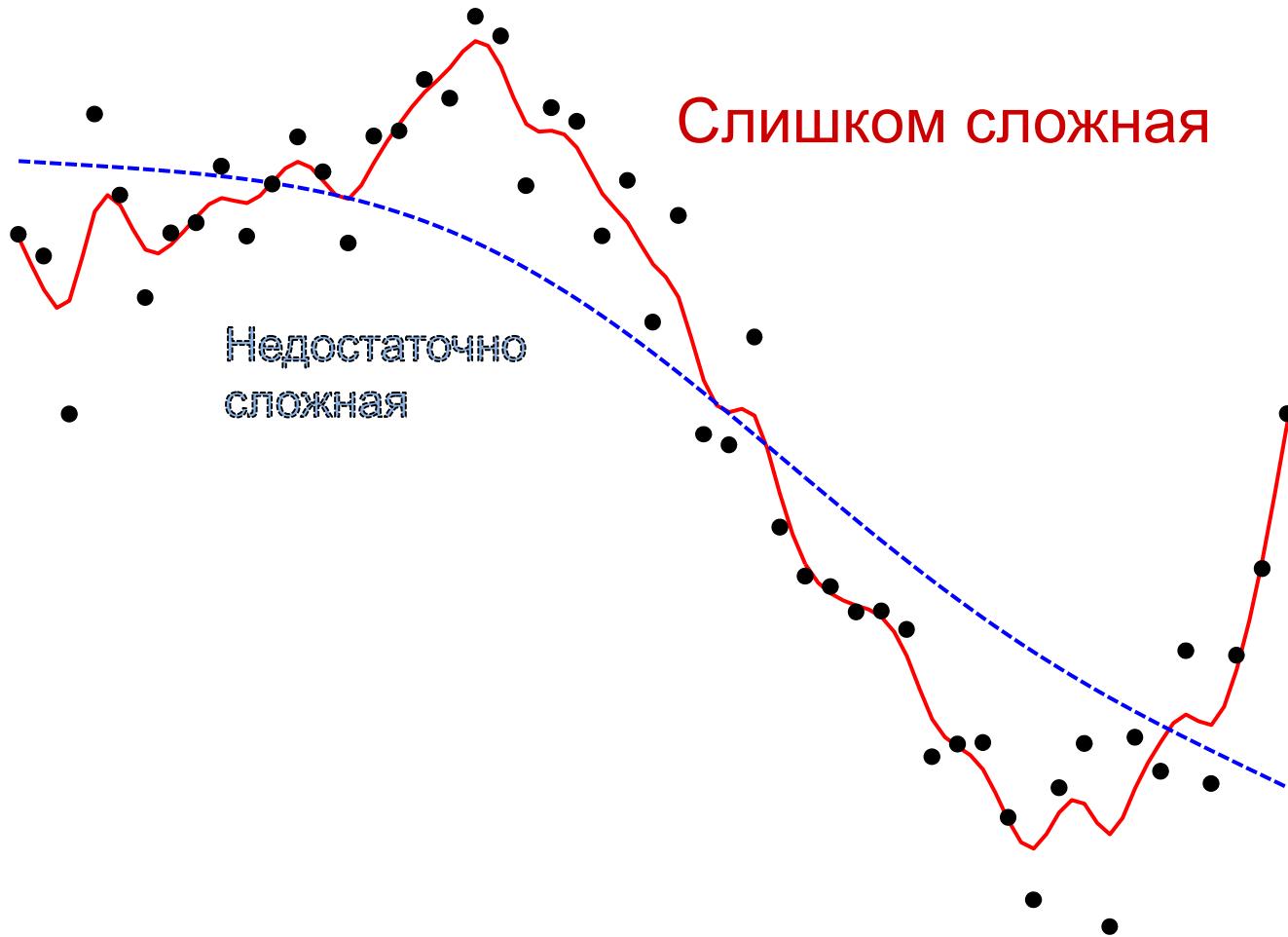


...

Сложность модели



Сложность модели



Сложность модели и переобучение

- Основная проблема методов машинного обучения!!!
- По сути:
 - Высокая точность на тренировочном наборе и плохая на тестовом
- Причины:
 - Сложность модели: например, для параметрических моделей много степеней свободы (параметров модели) или слишком сложное уравнение
 - Шум и выбросы в тренировочной выборке
 - Малый объем или неравномерность тренировочной выборки
- Обобщающая способность:
 - способность модели правильно прогнозировать «отклик» для объектов и ситуаций, которых не было в тренировочном наборе
 - метод называется состоятельным, если он с большой вероятностью делает маленькую ошибку на данных, которых не было в обучающей выборке

Как оценить?

MSE декомпозиция

$$\begin{aligned}MSE &= E[(\hat{D} - D)^2] = E[\hat{D}^2] + E[D^2] - E[2\hat{D}D] = \\&= \textcolor{blue}{Var(\hat{D})} + \textcolor{red}{Var(D)} + \textcolor{green}{(E[\hat{D}] - E[D])^2}\end{aligned}$$

↑

Дисперсия оценки Квадрат смещения

Дисперсия шума (не
зависит от модели)

Компромисс: Дисперсией vs Смещение!!!!

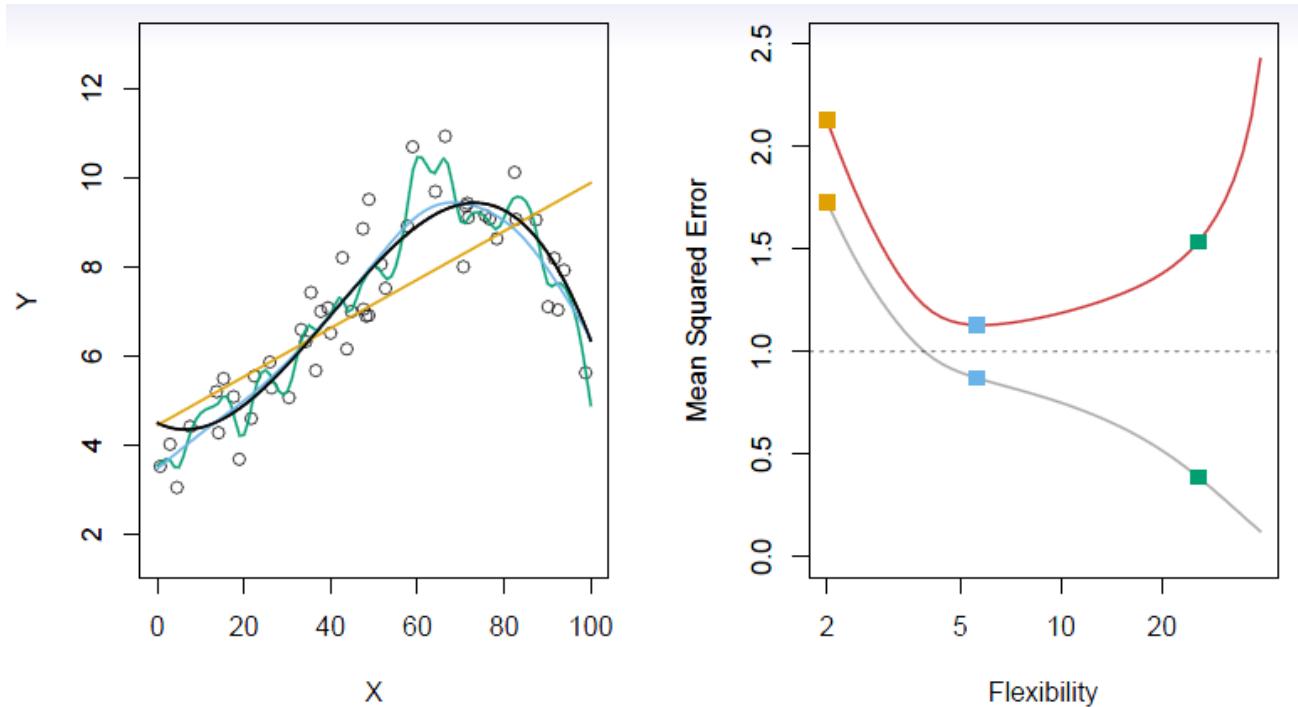
Сложнее модель => точнее приближение => меньше смещение +++

Сложнее модель => больше параметров => больше дисперсия ---

... и наоборот ...

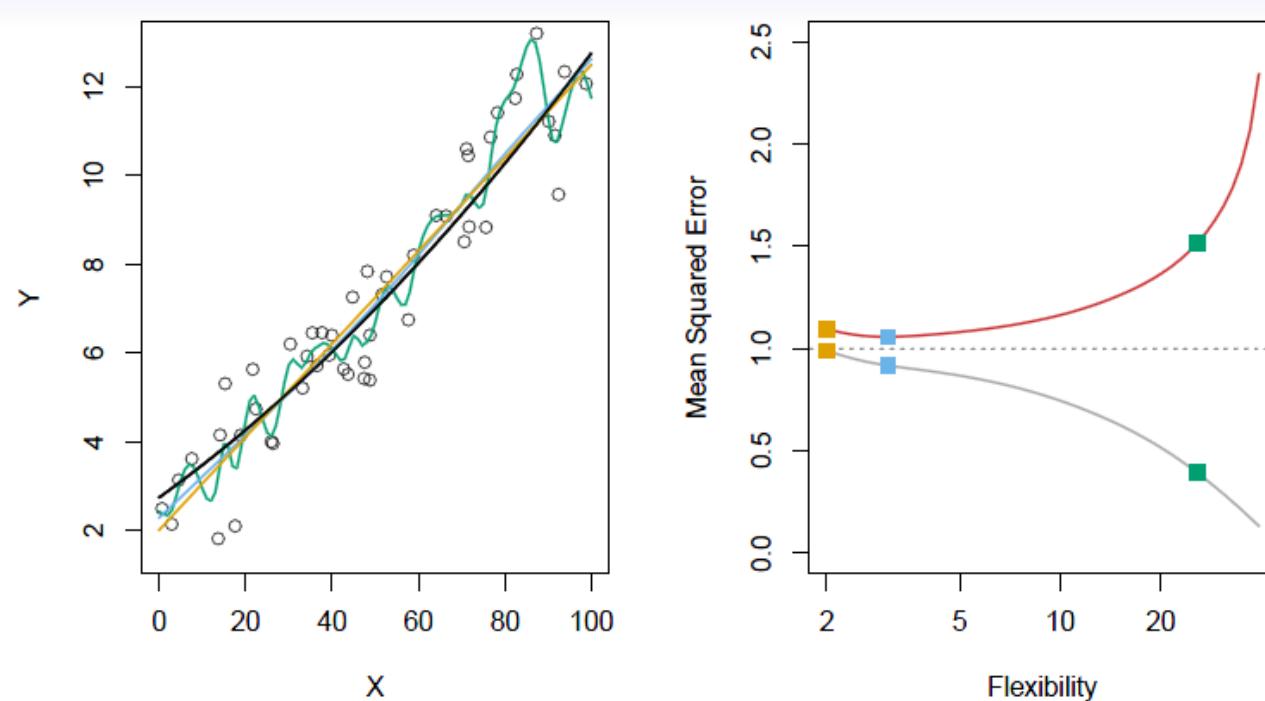
Поиск баланса между точностью и сложностью = поиск компромисса между
смещением и дисперсией

Оценка качества модели (сложная зависимость, много шума)



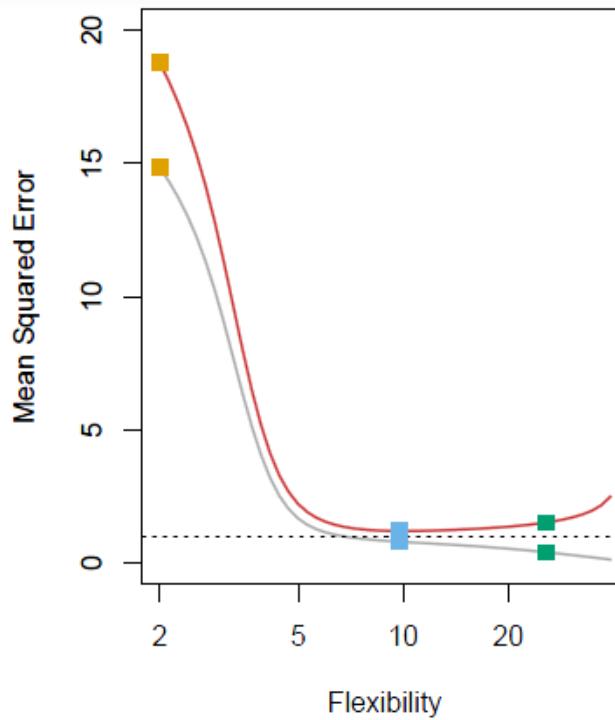
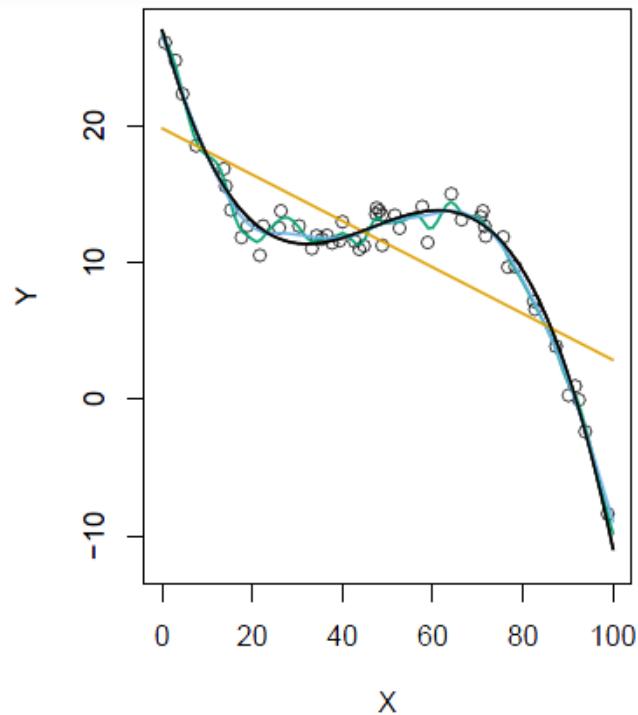
- Кривая, обозначенная черным цветом - истинные значения.
- Красная кривая на правом рисунке – MSE_{Te} , серая кривая – MSE_{Tr} .
- Оранжевая, голубая и зеленая кривые соответствуют подгонке моделей различной гибкости.
- Простые модели недообучены, сложные модели переобучены

Оценка качества модели (простая зависимость, много шума)



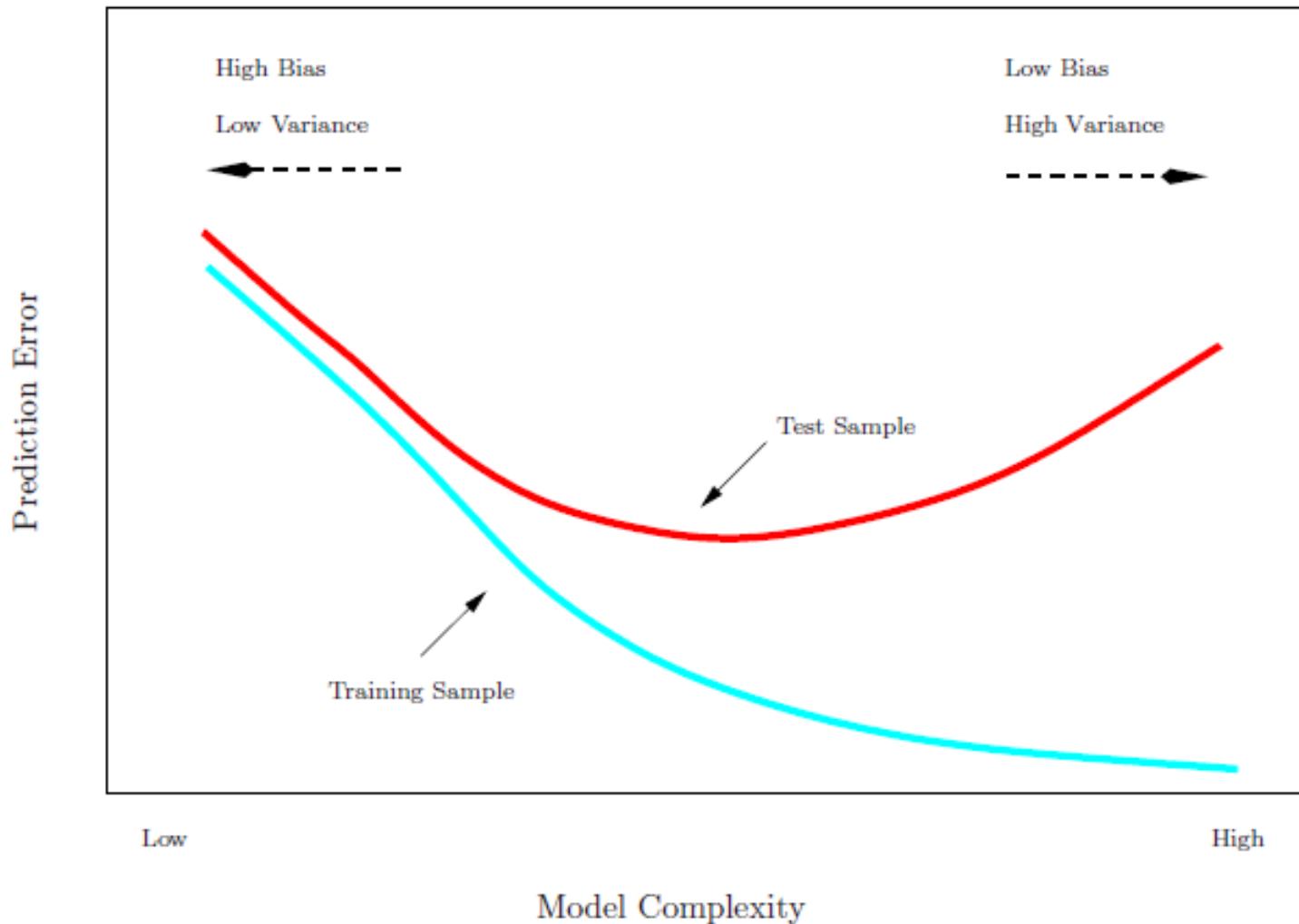
- Простые модели дают высокую обобщающую способность
- Сложные модели переобучены

Оценка качества модели (сложная зависимость, мало шума)

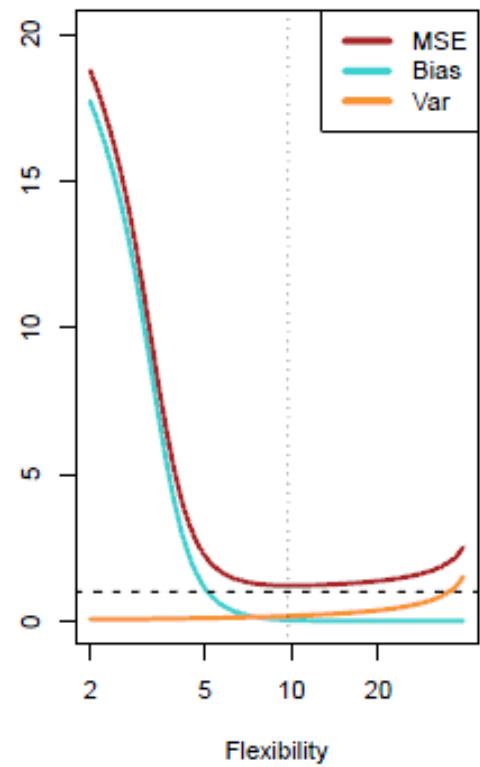
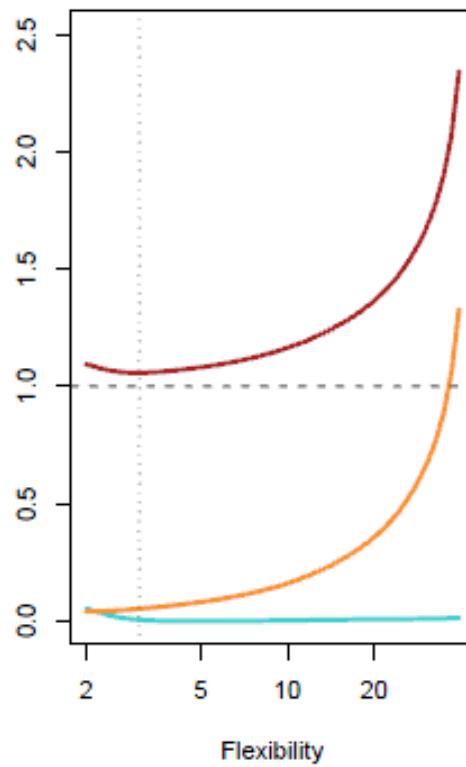
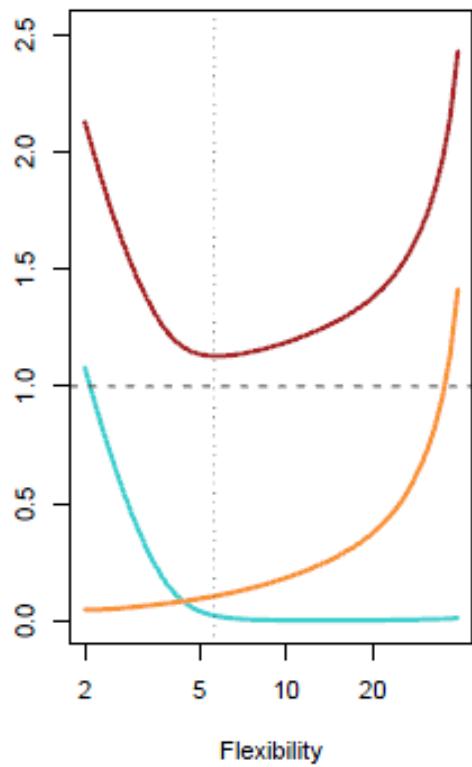


- Простые модели недообучены
- Сложные обладают хорошей обобщающей способностью

Качество на обучающем и тестовом наборе



Компромисс отклонения смещения для трех примеров



Валидация, кросс-валидация и бутстреппинг

- Эти методы позволяют:
 - оценить ошибки прогнозирования тестового набора
 - стандартное отклонение и смещения оценок параметров модели (бутстреппинг и кросс-валидация)
 - выбрать лучшую модель
- Различия между *ошибкой тестирования* и *ошибкой обучения*:
 - Ошибка тестирования - это усредненная ошибка, которая возникает в результате применения метода статистического обучения для прогнозирования отклика на новом наблюдении, которое не было задействовано в процессе обучения.
 - Ошибка обучения вычисляется после применения метода статистического обучения к наблюдениям, используемым в обучении.

Применение валидационного набора

- Разделим случайным образом имеющийся набор образцов на две части: *обучающую и валидационную выборки*.



- Построим модель на обучающем наборе и используем ее для прогнозирования откликов наблюдений в валидационном наборе.
- Полученная ошибка на валидационном множестве дает оценку тестовой ошибки..

Использование валидационного набора данных

		<i>Training Data</i>						<i>Validation Data</i>					
		<i>inputs</i>				<i>target</i>				<i>inputs</i>		<i>target</i>	
		1	2	3	4			1	2	3	4		
1		1	1	1	1	1		1	1	1	1	1	
2		1	1	1	1	2		1	1	1	1	2	
3		1	1	1	1	3		1	1	1	1	3	
4		1	1	1	1	4		1	1	1	1	4	
5		1	1	1	1	5		1	1	1	1	5	

Модель строится на тренировочном наборе, а
оценивается на валидационном

Оценка моделей

Training Data

	<i>inputs</i>			<i>target</i>

Validation Data

	<i>inputs</i>			<i>target</i>



Сложность модели Валидационная оценка



Оценка качества моделей на валидационном наборе

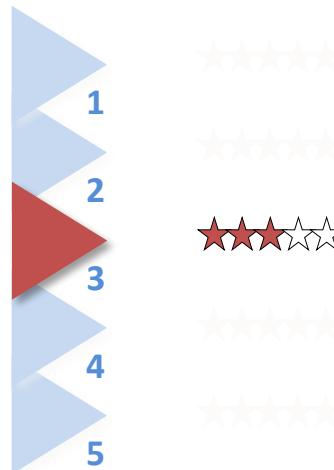
Выбор модели

Training Data

	<i>inputs</i>			<i>target</i>

Validation Data

	<i>inputs</i>			<i>target</i>



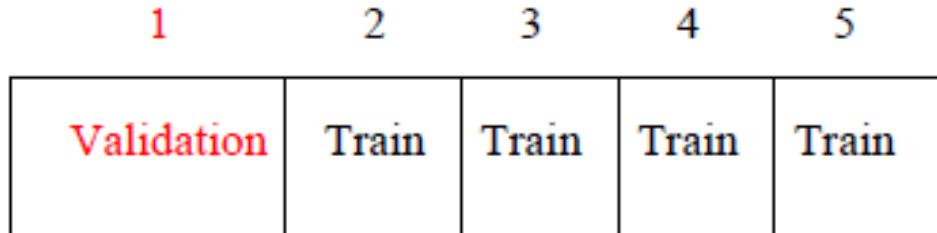
Самая простая модель среди
самых лучших на
валидационном наборе

Недостатки подхода применения валидационного набора

- Если плохое разбиение:
 - Валидационная оценка ошибки тестирования может сильно варьироваться в зависимости от того, какие именно наблюдения включены в обучающий наборе, а какие в валидационный.
- Не вся информация используется при обучении:
 - При валидационный подходе только подмножество наблюдений (те, которые включены в обучающий наборе, а не в валидационный) используются для построения модели.
- Чрезмерный оптимизм:
 - Ошибка на валидационном наборе может иметь тенденцию переоценивать ошибку тестирования

Кросс-валидация

- Широко используемый подход для оценки ошибки тестирования.
- Оценки могут быть использованы для:
 - выбора оптимальной модели,
 - оценки тестовой ошибки результирующей выбранной модели.
- Идея - разделить данные на K частей равного размера. Мы удаляем часть k , строим модель на оставшихся частях, а затем получаем прогнозы для удаленной k -ой части.



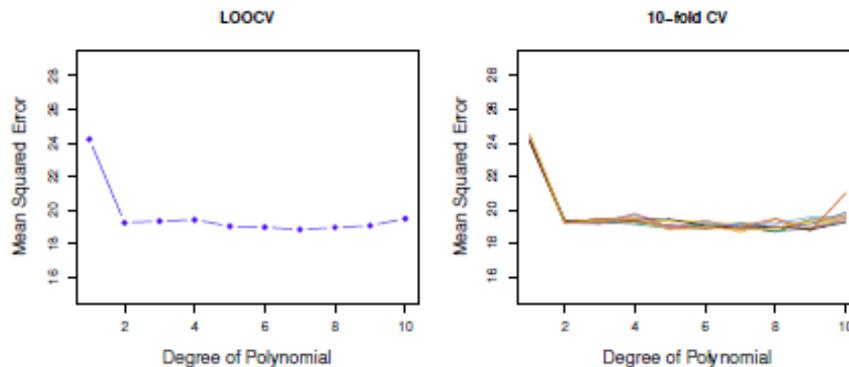
- Это делается в свою очередь для каждой части $k = 1, 2, \dots, K$, а затем результаты объединяются.

Кросс-валидация для оценки ошибки

- Обозначим K частей как C_1, C_2, \dots, C_K , где C_k - это индексы наблюдений в части k . Есть n_k наблюдения в части k : если N кратно K , то $n_k = n/K$.
- Вычислим

$$CV_{(K)} = \sum_{k=1}^K \frac{n_k}{n} MSE_k$$

где $MSE_k = \sum_{i \in C_k} (y_i - \hat{y}_i)^2 / n_k$ \hat{y}_i - подгонка для наблюдения i , полученная на данных с удаленной частью k .

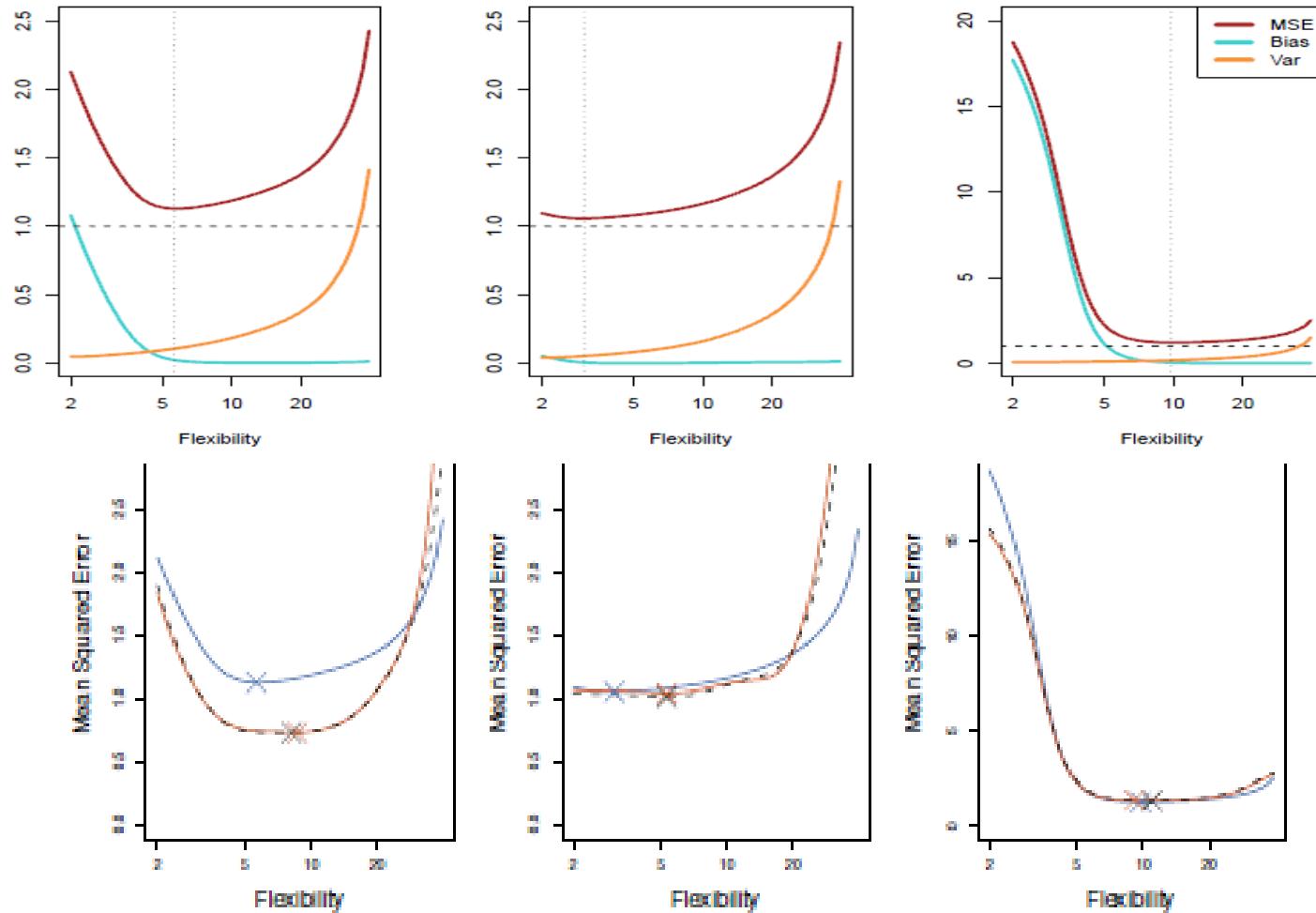


- При $K = n$ имеем n папок или кросс-валидацию с попаренным исключением одной из частей (*leave-one out cross-validation, LOOCV*).

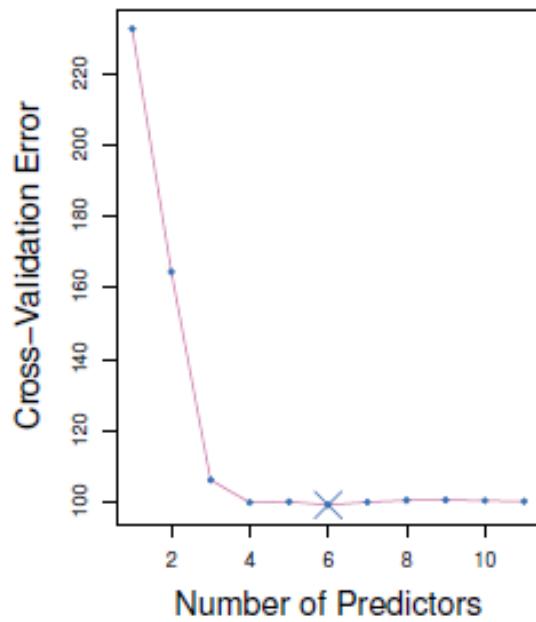
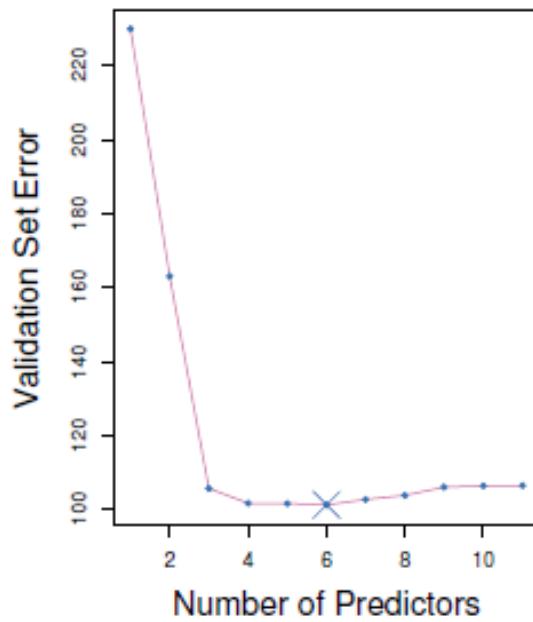
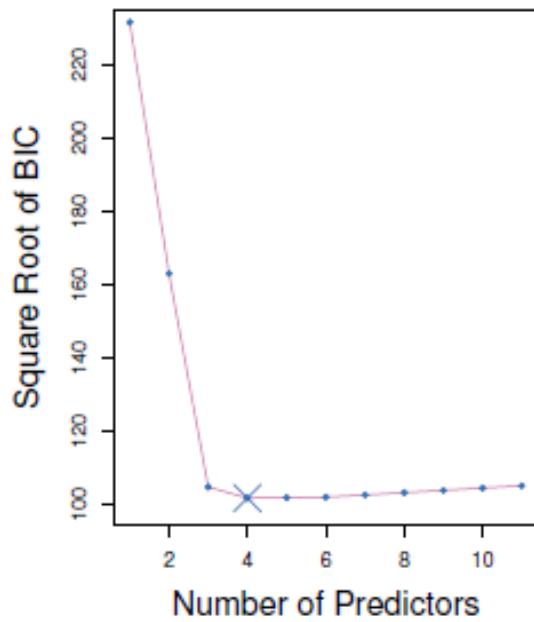
Кросс-валидация для оценки метапараметров и выбора модели

- Зачастую кросс-валидацию используют не для оценки ошибки, а для выбора метапараметров
 - Запускают кросс-валидацию для разных значений метапараметров
 - Рассчитывают кросс-валидационные ошибки для каждого варианта
 - Выбирают лучшее значение метапараметра по кросс-валидационной ошибке
 - Перестраивают модель на всей выборке с этим значением метапараметра

Кросс-валидация для оценки метапараметров модели



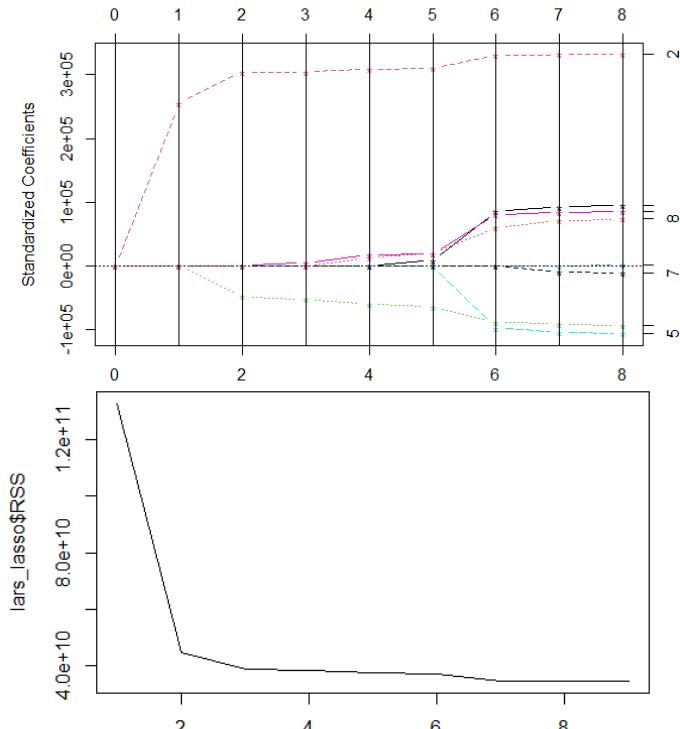
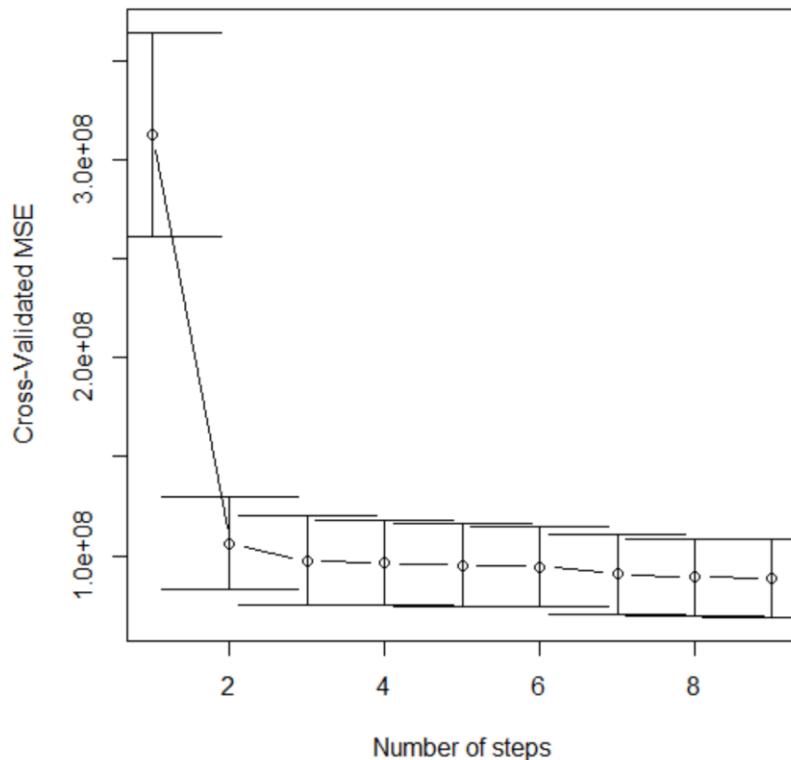
Пример



“Встроенная” кросс-валидация в процедуры в R

- Многие процедуры построения моделей в R имеют возможности по отбору гиперпараметров и по оценке ошибок с помощью кросс-валидации, например, `plsR` (для оценки числа компонент) или `Tsp` (для оценки штрафа), `cv.lars` для выбора оптимального шага.

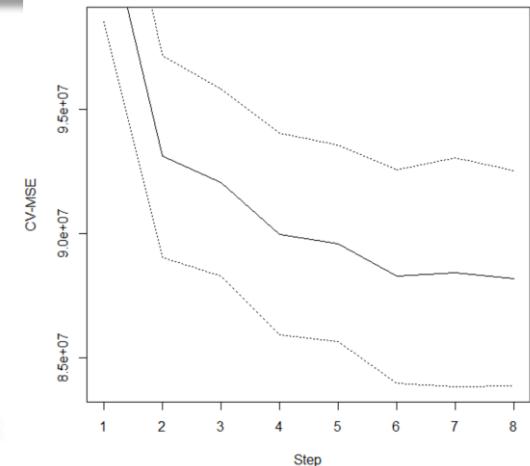
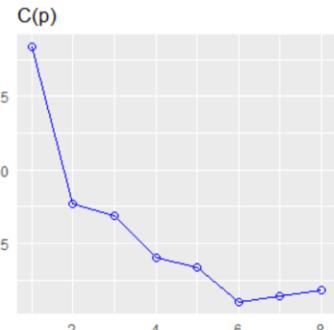
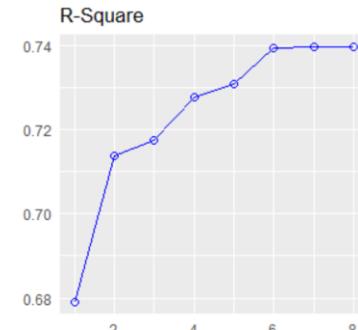
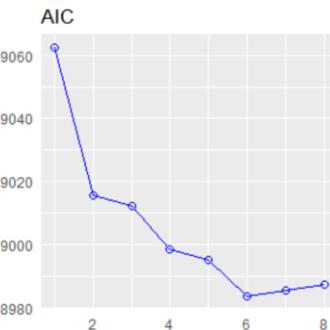
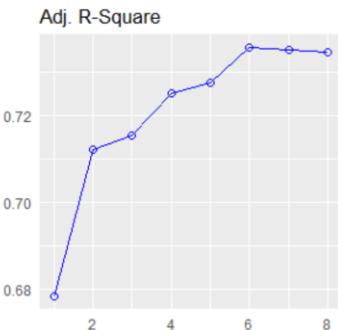
```
> lasso<-cv.lars(as.matrix(cars[,c("Weight", "Horsepower", "Wheelbase", "Length",
+ "EngineSize", "Cylinders", "MPG_City", "MPG_Highway")]),cars$Inv, K=25,mode="step")
```



Общие процедуры кросс-валидации

```
crossval(predfun, X, Y, K=10, B=20, verbose=TRUE, ...)  
predfun(Xtrain, Ytrain, Xtest, Ytest, ...)
```

```
> predfun.glm = function(train.x, train.y, test.x, test.y)  
{  
+   glm.fit = glm(train.y ~ . , data=train.x,family=Gamma(link=log))  
+   ynew = predict.glm(glm.fit, test.x,type = "response")  
+   out = mean( (ynew - test.y)^2 )  
+   return( out )  
+ }  
> lm_model<-lm(cars$Inv~.,cars[,c("Weight", "Horsepower", "Wheelbase",  
+ "Length", "EngineSize", "Cylinders", "MPG_City", "MPG_Highway")])  
> k<-ols_step_best_subset(lm_model,metric="AIC")  
> stat<-c()  
> se<-c()  
> for (p in k$predictors) {  
+   crs<-crossval(predfun.lm,data.frame(cars[,strsplit(p," ")[[1]]]),cars$Inv)  
+   stat<-append(stat,crs$stat)  
+   se<-append(se,crs$stat.se)  
+ }  
  
> plot(k)
```



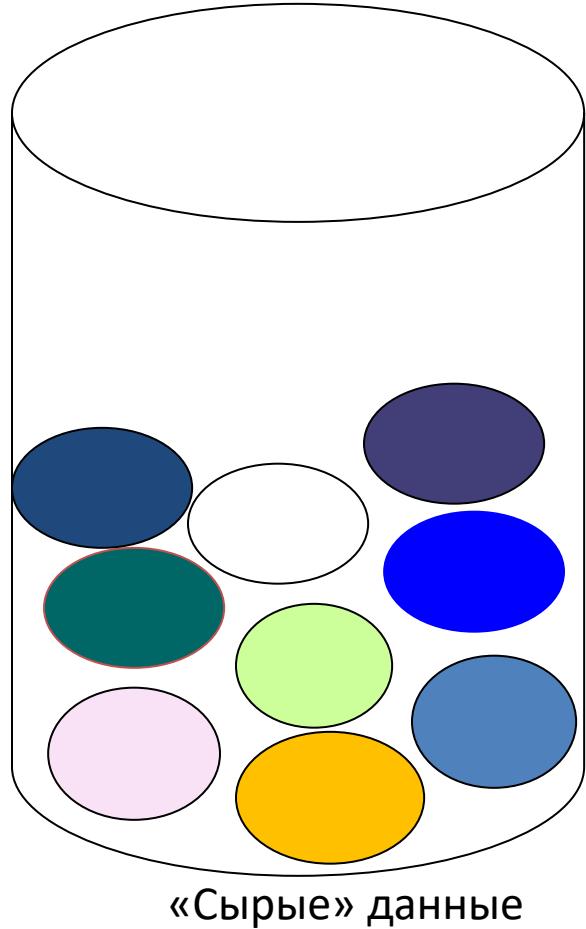
```
> plot(x=seq(1,length(k$predictors)),y=stat-se,  
+       ylab="CV-MSE",xlab="Step",type="l",lty=3)  
> lines(x=seq(1,length(k$predictors)),y=stat+se,type="l", lty=3)  
> lines(x=seq(1,length(k$predictors)),y=stat,type="l")|
```

Бутстрепинг

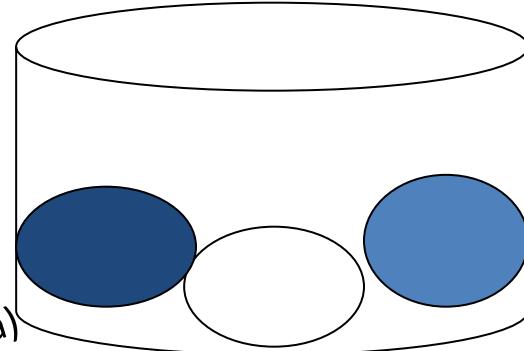
- Бутстрепинг представляет собой гибкий и мощный статистический инструмент, который может быть использован для количественной оценки неопределенности, связанной с данным методом статистического обучения.
- Например, он может позволить произвести оценку стандартной ошибки коэффициента или доверительного интервала для этого коэффициента.
- «Есть мнение», что использование термина бутстрепинг происходит от фразы, чтобы *to pull oneself up by one's bootstraps*, - это мнение, основанное на цитате из книги «Удивительные приключения барона Мюнхгаузена»

Барон упал на дно глубокого озера. Когда казалось, что все было потеряно, он решил вытащить себя своими собственными силами.

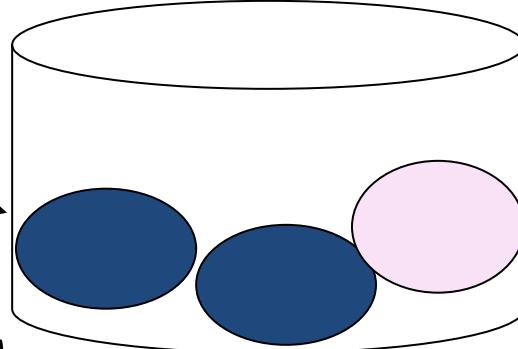
Случайная выборка с возвратом и без



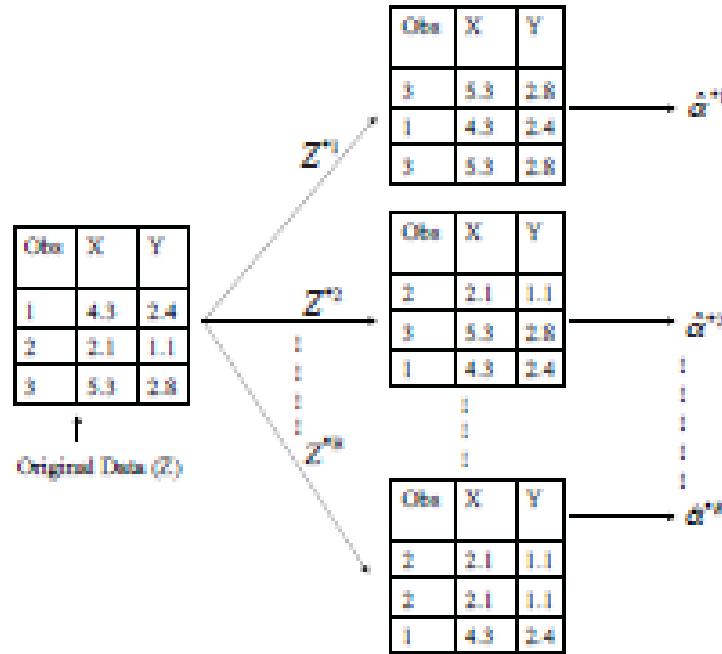
SRSWOR
(простая случайная
выборка без возврата)



SRSWR
(простая случайная
выборка с возвратом)



Демонстрационный пример с тремя наблюдениями



- Графическая иллюстрация бутреппингового подхода на маленькой выборке, содержащей $N = 3$ наблюдений.
- Каждый бутреппинговый набор данных содержит n наблюдений, отобранных с заменой из исходного набора данных.
- Каждый такой набор данных начальной используется для получения оценки α

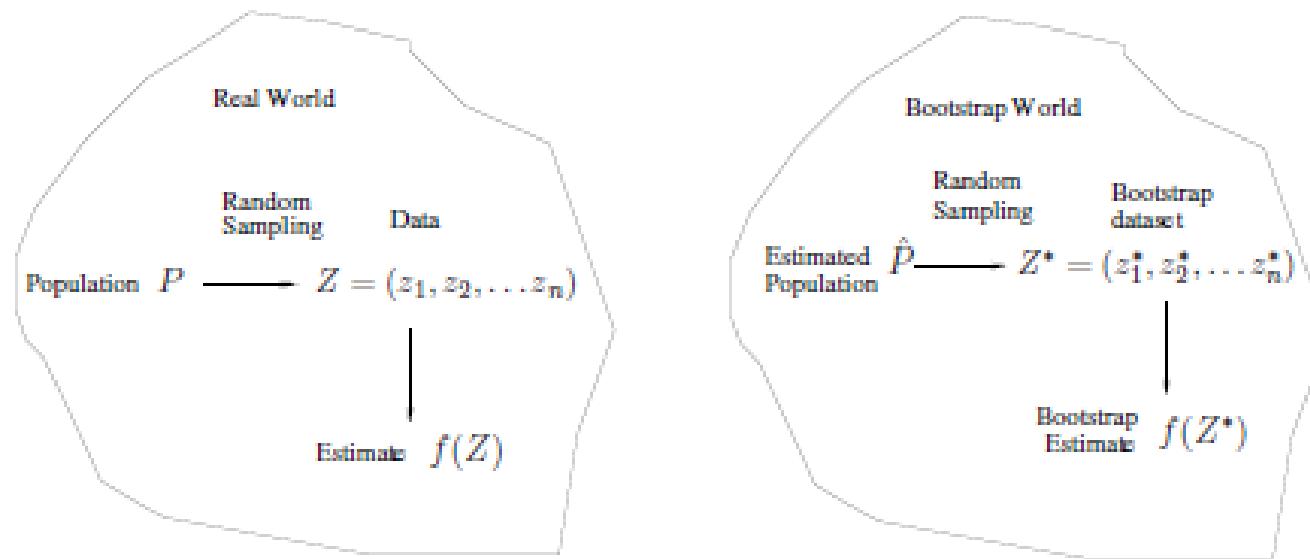
Бутстрепинг

- Обозначая первый набор данных бутстреппинга как Z^{*1} , мы используем Z^{*1} , чтобы выполнить новую оценку для α , которую обозначим $\hat{\alpha}^{*1}$
- Эта процедура повторяется B раз для некоторого большого значения B (например, 100 или 1000), чтобы получить B различных наборов данных бутстреппинга $Z^{*1}, Z^{*2}, \dots, Z^{*B}$ и B соответствующих оценок α : $\hat{\alpha}^{*1}, \hat{\alpha}^{*2}, \dots, \hat{\alpha}^{*B}$.
- Оценим стандартную ошибку этих оценок бутстреппинга, используя формулу:

$$SE_B(\hat{\alpha}) = \sqrt{\frac{1}{B-1} \sum_{r=1}^B (\hat{\alpha}^{*r} - \bar{\hat{\alpha}}^*)^2}.$$

- Она служит в качестве оценки стандартной ошибки, полученной на исходном наборе данных.

Общая схема бутстрепинга



- В более сложных ситуациях, определение подходящего способа для получения выборок бутстрепинга может потребовать значительных усилий.
- Например, если данные представляют собой временные ряды, мы не можем просто выбирать наблюдения с замещением

Общие процедуры бутстрепинга

```
boot(data, statistic, R, sim = "ordinary", stype = c("i", "f", "w"),
      strata = rep(1,n), L = NULL, m = 0, weights = NULL, ran.gen = function(d,
      p) d, mle = NULL, simple = FALSE, ..., parallel = c("no", "multicore",
      "snow"), ncpus =getOption("boot.ncpus", 1L), cl = NULL)
```

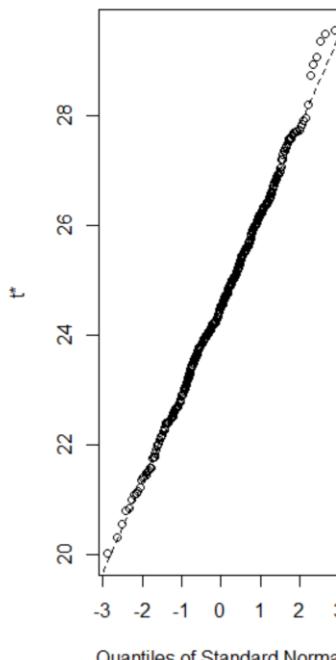
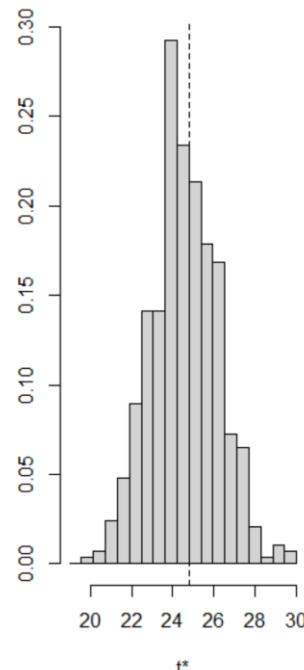
- Можно делать бутстреп оценки как одномерных статистик (ниже пример для статистики отклонения в гамма-регрессии) так и векторных (например, параметров регрессионных моделей) :

```
> dv<-function(formula, data, idxs){
+   return(glm(formula,data[idxs,],family=Gamma(link=log))$deviance)
+ }
>
> r<-boot(data=cars,R=500,statistic=dv,formula=Inv~Horsepower+Weight)
> boot.ci(r)
BOOTSTRAP CONFIDENCE INTERVAL CALCULATIONS
Based on 500 bootstrap replicates

CALL :
boot.ci(boot.out = r)

Intervals :
Level      Normal          Basic
95%  (21.85, 28.21 )  (21.88, 28.19 )

Level      Percentile        BCa
95%  (21.41, 27.72 )  (22.12, 29.06 )
Calculations and Intervals on Original Scale
Some BCa intervals may be unstable
Warning message:
In boot.ci(r) : bootstrap variances needed for studentized intervals
> plot(r)
```



Бутстреп оценка параметров «плохой» нелинейной модели

```
> cfs<-function(formula, data, idxs){  
+ return(coef(nls(formula,data=data[idxs,],control=nlc,algorithm="port")))  
+ }  
>  
> r<-boot(data=cars_std,R=100,statistic=cfs,formula=full_formula)  
> summary(model_nls)
```

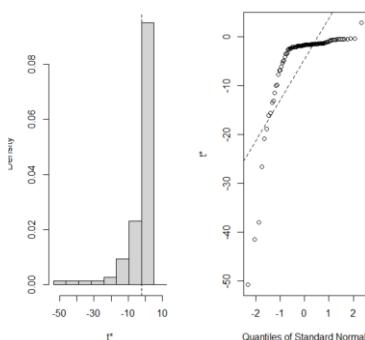
Formula: Invoice ~ exp(b0 + b1 * tanh(a1 + a11 * Weight + a12 * Horsepower)) +
b2 * tanh(a2 + a21 * Weight + a22 * Horsepower)

Parameters:

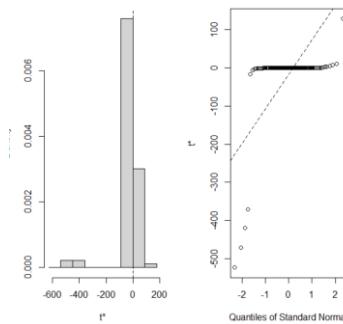
	Estimate	Std. Error	t value	Pr(> t)
b0	0.41461	0.25664	1.616	0.10695
b1	0.81263	0.30149	2.695	0.00731 **
a1	-16.79567	38.04382	-0.441	0.65909
a11	-2.02621	14.23181	-0.142	0.88685
a12	5.37150	14.16092	0.379	0.70464
b2	-2.13610	0.68919	-3.099	0.00207 **
a2	0.37602	0.11467	3.279	0.00113 **
a21	0.01435	0.01935	0.742	0.45865
a22	-0.41118	0.14460	-2.843	0.00468 **

Bootstrap Statistics :			
	original	bias	std. error
t1*	0.41460903	-5.89549462	43.88940
t2*	0.81262790	10.19033566	43.42281
t3*	-16.79566536	2.41897700	31.20815
t4*	-2.02621374	1.49746391	15.62944
t5*	5.37150218	0.05377063	15.56864
t6*	-2.13609952	-3.31719294	12.49244
t7*	0.37602282	6.15537056	32.08285
t8*	0.01435011	-11.98235803	92.56977
t9*	-0.41118156	-6.66007849	52.89654
>			

```
> plot(r,index=6)
```



```
> plot(r,index=8)
```



Пропущенные значения

- Импутация константным значением - все пропуски для переменной заменяются на:
 - Моду (для категориальных) или мат. ожидание, или пользовательскую константу или робастные оценки
- Импутация псевдослучайным значением:
 - В соответствии с распределением
- Импутация прогнозом (оценкой)
 - можно делать свои модели, но учитывать, что на входе могут быть пропуски

Для неслучайных пропусков – индикаторные переменные

- Одна на все наблюдение
- Своя для каждой переменной



Оценки
 $x_i = f(x_1, \dots, x_p)$

Пример для сравнения разных методов импутации

- «Испортим» данные для модели $\text{Inv} \sim \text{Weight} + \text{Length} + \text{Horsepower}$

```
> for(i in seq(1,length(cars_miss[,1]))){  
+ if(rand()<0.1) cars_miss[i,]$Weight=NA  
+ if(rand()<0.05) cars_miss[i,]$Horsepower=NA  
+ if(rand()<0.01) cars_miss[i,]$Length=NA  
+ }  
> cars_miss  
    Inv Weight Length Horsepower  
1   33337    4451    189      265  
2   21761      NA     172      200  
3   24647    3230    183      200  
4   30299    3575    186      270  
5   39014    3880    197      NA  
6   41100    3893    197      225  
7   79978    3153    174      290  
8   23508      NA     179      170  
9   32506    3638    180      170  
  
> cars_miss$X_Weight<-ifelse(is.na(cars_miss$Weight),1,0)  
> cars_miss$X_Length<-ifelse(is.na(cars_miss$Length),1,0)  
> cars_miss$X_Horsepower<-ifelse(is.na(cars_miss$Horsepower),1,0)  
> cars_miss  
    Inv Weight Length Horsepower X_Weight X_Length X_Horsepower  
1   33337    4451    189      265      0       0       0  
2   21761      NA     172      200      1       0       0  
3   24647    3230    183      200      0       0       0  
4   30299    3575    186      270      0       0       0  
5   39014    3880    197      NA       0       0       1  
6   41100    3893    197      225      0       0       0  
7   79978    3153    174      290      0       0       0  
8   23508      NA     179      170      1       0       0  
9   32506    3638    180      170      0       0       0  
10  28846    3462    179      220      0       0       0  
11  28846    3462    179      220      0       0       0
```

```
> predfun.lm = function(train.x, train.y, test.x, test.y){  
+ {  
+   lm.fit = lm(train.y ~ . , data=train.x)  
+   ynew = predict(lm.fit, test.x )  
+   out = mean( (ynew - test.y)^2 )  
+   return( out )  
+ }  
>  
> cv<-crossval(predfun.lm, cml[,-1], cml$Inv, K=10)
```

Модель без подстановки

```
> m<-lm(Inv~Weight+Length+Horsepower,data=cars_miss)
> summary(m)

Call:
lm(formula = Inv ~ Weight + Length + Horsepower, data = cars_miss)

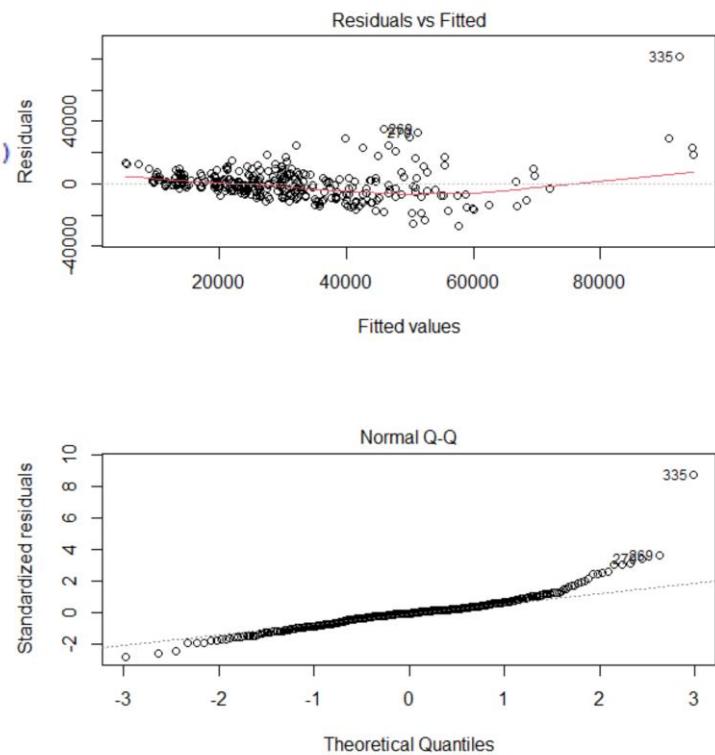
Residuals:
    Min      1Q  Median      3Q     Max 
-26915 -4937   -475   3520  81157 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 23549.079   7271.082   3.239  0.00132 ** 
Weight       -0.505     1.133   -0.446  0.65600    
Length      -218.327    48.761   -4.478 1.02e-05 *** 
Horsepower    227.762    9.708   23.462 < 2e-16 *** 
---
Signif. codes:  0 '****' 0.001 '***' 0.01 '**' 0.05 '.' 0.1 ' ' 1

Residual standard error: 9733 on 348 degrees of freedom
(74 observations deleted due to missingness)
Multiple R-squared:  0.7072,    Adjusted R-squared:  0.7047 
F-statistic: 280.2 on 3 and 348 DF,  p-value: < 2.2e-16

> plot(m)
```

$$CV (MSE) = 100671834 \pm 4955341$$



Подстановка константной характеристики распределения

```
> cml<-cars_miss  
> cml$Weight[cml$X_Weight==1]<-mean(cars_miss$Weight,na.rm = TRUE)  
> cml$Length[cml$X_Length==1]<-mean(cars_miss$Length,na.rm = TRUE)  
> cml$Horsepower[cml$X_Horsepower==1]<-mean(cars_miss$Horsepower,na.rm = TRUE)  
> |  
> ml<-lm(Inv~Weight+Length+Horsepower+X_Weight+X_Length+X_Horsepower,data=cml)  
> summary(ml)
```

```
Call:  
lm(formula = Inv ~ Weight + Length + Horsepower + X_Weight +  
    X_Length + X_Horsepower, data = cml)
```

Residuals:

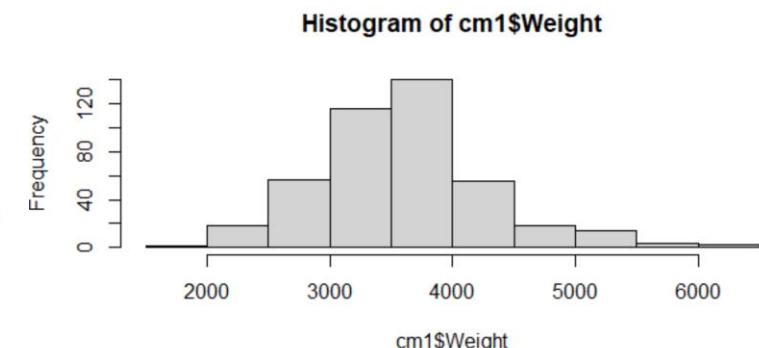
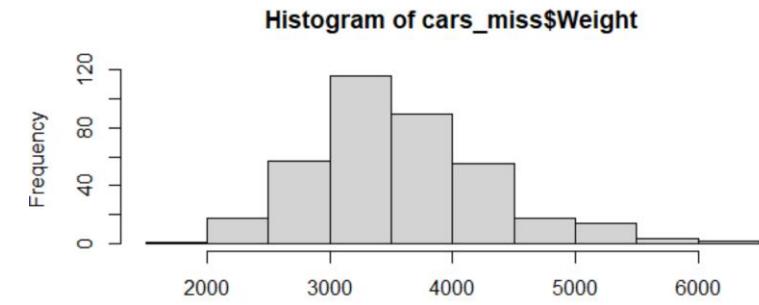
Min	1Q	Median	3Q	Max
-25335	-5457	-217	3654	85416

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	22023.255	6869.541	3.206	0.00145 **
Weight	1.051	1.067	0.984	0.32559
Length	-224.885	45.268	-4.968	9.88e-07 ***
Horsepower	214.227	9.082	23.587	< 2e-16 ***
X_Weight	-248.274	1534.097	-0.162	0.87151
X_Length	413.811	5210.173	0.079	0.93673
X_Horsepower	2440.942	2317.006	1.053	0.29272

Signif. codes:	0 ****	0.001 ***	0.01 **	0.05 *
	'.'	0.1 '	'	1

```
Residual standard error: 10340 on 419 degrees of freedom  
Multiple R-squared:  0.6626,   Adjusted R-squared:  0.6578  
F-statistic: 137.1 on 6 and 419 DF,  p-value: < 2.2e-16
```



CV (MSE) = 113356226 ± 4907095

Подстановка случайного значения эмпирического распределения

```
> cm2<-cars_miss
> for(i in seq(1,length(cm2[,1]))){
+ if (cm2$X_Weight[i]==1) cm2$Weight[i]<-remp(l,as.vector(na.omit(cars_miss$Weight*1.0)))
+ if (cm2$X_Length[i]==1) cm2$Length[i]<-remp(l,as.vector(na.omit(cars_miss$Length*1.0)))
+ if (cm2$X_Horsepower[i]==1) cm2$Horsepower[i]<-remp(l,as.vector(na.omit(cars_miss$Horsepower*1.0)))
+ }
> m2<-lm(Inv~Weight+Length+Horsepower+X_Weight+X_Length+X_Horsepower,data=cm2)
> summary(m2)
```

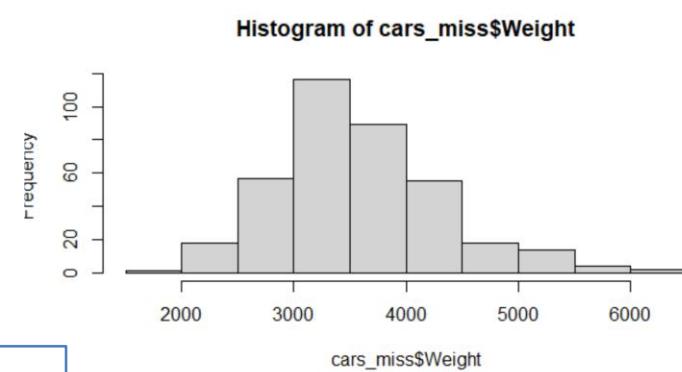
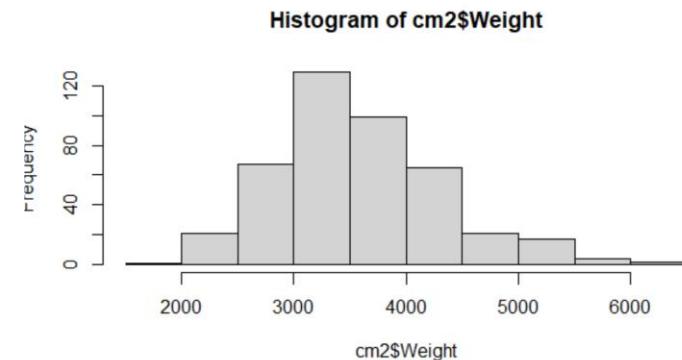
Call:
lm(formula = Inv ~ Weight + Length + Horsepower + X_Weight +
X_Length + X_Horsepower, data = cm2)

Residuals:
Min 1Q Median 3Q Max
-31371 -5345 -405 3451 88082

Coefficients:
Estimate Std. Error t value Pr(>|t|)
(Intercept) 20640.2316 6971.9513 2.960 0.00325 **
Weight 1.3171 0.9318 1.414 0.15825
Length -211.8355 44.2347 -4.789 2.33e-06 ***
Horsepower 205.0003 8.7976 23.302 < 2e-16 ***
X_Weight -473.9993 1579.6637 -0.300 0.76428
X_Length 65.3492 5357.8925 0.012 0.99027
X_Horsepower -109.9614 2393.9030 -0.046 0.96338

Signif. codes: 0 '****' 0.001 '***' 0.01 '**' 0.05 '*' 0.1 '.' 1

Residual standard error: 10650 on 419 degrees of freedom
Multiple R-squared: 0.6424, Adjusted R-squared: 0.6372
F-statistic: 125.4 on 6 and 419 DF, p-value: < 2.2e-16



$$CV(MSE) = 120200281 \pm 4989768$$

Подстановка на основе оценки

```
> m_h<-lm(Horsepower~Weight+Length,data=cars)
> m_w<-lm(Weight~Horsepower+Length,data=cars)
> m_l<-lm(Length~Weight+Horsepower,data=cars)
> cm3<-cm3
> for(i in seq(1,length(cm3[,1]))){
+ if (cm3$X_Weight[i]==1) cm3$Weight[i]<-predict(m_w,list(Horsepower=cm3$Horsepower[i],Length=cm3$Length[i]))
+ if (cm3$X_Length[i]==1) cm3$Length[i]<-predict(m_l,list(Horsepower=cm3$Horsepower[i],Weight=cm3$Weight[i]))
+ if (cm3$X_Horsepower[i]==1) cm3$Horsepower[i]<-predict(m_h,list(Weight=cm3$Weight[i],Length=cm3$Length[i]))
+ }
> summary(m3)
```

Call:
lm(formula = Inv ~ Weight + Length + Horsepower + X_Weight +
X_Length + X_Horsepower, data = cm3)

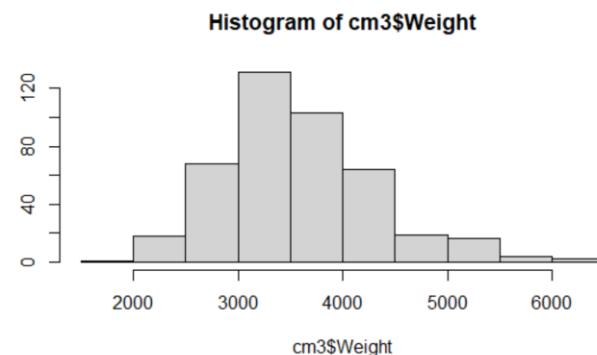
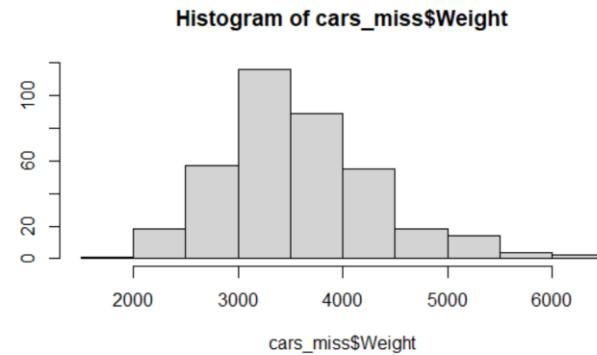
Residuals:
Min 1Q Median 3Q Max
-26247 -5470 -473 3365 82721

Coefficients:

	Estimate	Std. Error	t value	Pr(> t)
(Intercept)	22997.7653	7031.0127	3.271	0.00116 **
Weight	-0.1957	1.1618	-0.168	0.86631
Length	-216.1354	47.9713	-4.506	8.6e-06 ***
Horsepower	222.8051	9.5595	23.307	< 2e-16 ***
X_Weight	-9.0310	1494.3153	-0.006	0.99518
X_Length	-509.6671	5073.1725	-0.100	0.92002
X_Horsepower	3768.6082	2259.1315	1.668	0.09603 .

Signif. codes: 0 '****' 0.001 '***' 0.01 '**' 0.05 '*' 0.1 '.' 0.1 ' ' 1

Residual standard error: 10080 on 419 degrees of freedom
Multiple R-squared: 0.6794, Adjusted R-squared: 0.6748
F-statistic: 148 on 6 and 419 DF, p-value: < 2.2e-16



$$CV(MSE) = 107589107 \pm 4668721$$

Множественная подстановка

```
mice(data, m = 5, method = NULL,...)  
with(data, exp,...)  
pool(object, dfcom = NULL, rule = NULL)
```

- Общая схема:

1. mice - генерируем m наборов с подстановкой по выбранному методу
2. with - строим m моделей
3. pool - результат усредняем для точечных оценок и рассчитываем доверительные интервалы, если нужно

```
> head(fit)
$call
with.mids(data = imp, expr = lm(Inv ~ Weight + Length + Horsepower +
    X_Weight + X_Length + X_Horsepower))

$call1
mice(data = cars_miss)

$nmis
      Inv      Weight      Length      Horsepower      X_Weight      X_Length      X_Horsepower
          0           52            4             21              0                 0                  0                  0

$analyses
$analyses[[1]]

Call:
lm(formula = Inv ~ Weight + Length + Horsepower + X_Weight +
    X_Length + X_Horsepower)

Coefficients:
(Intercept)      Weight      Length      Horsepower      X_Weight      X_Length      X_Horsepower
 23920.0603     0.1366   -225.9668     221.8211    -576.1811   -2065.0578    -430.5118

$analyses[[2]]

Call:
lm(formula = Inv ~ Weight + Length + Horsepower + X_Weight +
    X_Length + X_Horsepower)

Coefficients:
(Intercept)      Weight      Length      Horsepower      X_Weight      X_Length      X_Horsepower
 24182.2695     0.1326   -223.9623     218.8452    -436.7179   -1191.8556    1455.8644

> imp <- mice(cars_miss)           $analyses[[3]]
> fit <- with(data = imp, exp = lm(Inv~Weight+Length+Horsepower+X_Weight+X_Length+X_Horsepower))
> p<-pool(fit)
```

Множественная подстановка

```
> summary(imp)
Class: mids
Number of multiple imputations:  5
Imputation methods:
      Inv      Weight      Length Horsepower      X_Weight      X_Length X_Horsepower
      ""       "pmm"     "pmm"      "pmm"        ""          ""          ""
PredictorMatrix:
      Inv Weight Length Horsepower X_Weight X_Length X_Horsepower
Inv      0      1      1      1      1      1      1
Weight    1      0      1      1      1      1      1
Length    1      1      0      1      1      1      1
Horsepower 1      1      1      0      1      1      1
X_Weight   1      1      1      1      0      1      1
X_Length   1      1      1      1      1      0      1
Number of logged events:  75
      it im      dep meth      out
1 1 1      Weight pmm      X_Weight
2 1 1      Length pmm      X_Length
3 1 1 Horsepower pmm X_Horsepower
4 1 2      Weight pmm      X_Weight
5 1 2      Length pmm      X_Length
6 1 2 Horsepower pmm X_Horsepower

> > summary(p)
      term      estimate std.error statistic      df      p.value
1 (Intercept) 23163.2982238 6998.49692 3.3097533 245.66943 1.073935e-03
2      Weight -0.3079915  1.41645 -0.2174390 18.32296 8.302692e-01
3      Length -215.2037211 50.65211 -4.2486621 73.93618 6.187531e-05
4 Horsepower  223.4596107 10.34699 21.5965769 51.71809 0.000000e+00
5      X_Weight -633.0693300 1454.26400 -0.4353194 409.27028 6.635601e-01
6      X_Length -950.9738875 5113.66113 -0.1859673 241.96147 8.526262e-01
7 X_Horsepower 2170.2416017 3107.51995  0.6983838 14.56121 4.959403e-01
> |
```

$$CV(MSE) = 105801483 \pm 3969226$$

Сравнение разных методов импутации

